TRABAJO DE FIN DE CARRERA

TÍTULO DEL TFC: VirtualEPSC, el mundo virtual 2.0 del Campus del Baix Llobregat

TITULACIÓN: Ingeniería Técnica de Telecomunicación, especialidad en Telemática

AUTORES: Luis Miguel Amorós Martínez
Noemí Arbós Linio

DIRECTOR: Toni Oller Arcas

FECHA: 14 de julio de 2010

Título: VirtualEPSC, el mundo virtual 2.0 del Campus del Baix Llobregat
Autores: Luis Miguel Amorós Martínez
Noemí Arbós Linio
Director: Toni Oller Arcas
Fecha: 14 de julio de 2010

Resumen

En las últimas décadas, se han producido cambios tecnológicos de gran envergadura que han provocado una ruptura brusca con las tecnologías existentes hasta el momento. Una de las más importantes es Internet, que se ha convertido en el entorno de comunicación más importante de la historia, con más de mil millones de usuarios en todo el mundo.

Internet ha sufrido una serie de cambios en los últimos años y uno de los últimos conceptos que han surgido es el de web 2.0. Esta filosofía se basa en dar un rol más activo a los usuarios, por ejemplo, utilizando la colaboración de los usuarios en Internet, también llamado inteligencia colectiva; y en el diseño centrado en el usuario.

Este trabajo describe los pasos seguidos para desarrollar una web 2.0 que aloja un mundo virtual que emula el Campus del Baix Llobregat. Se describe cómo son el diseño y la arquitectura del proyecto, y cómo se ha hecho la implementación de las diferentes partes.

El resultado se denomina VirtualEPSC, una web 2.0 que aloja una aplicación multimedia en 2D que emula el Campus del Baix Llobregat, donde los usuarios podrán interactuar entre ellos. Esta plataforma permite promocionar el campus de una manera divertida y que las personas que quieran conocerlo puedan visitarlo de una manera virtual. Además, en la aplicación se ofrecen servicios relacionados con el campus y se proporciona la posibilidad de que usuarios registrados puedan crear sus propias zonas dentro del mundo virtual, incentivando aún más la participación de los usuarios.

Title: VirtualEPSC, the virtual world 2.0 of Campus del Baix Llobregat
Authors: Luis Miguel Amorós Martínez
Noemí Arbós Linio
Director: Toni Oller Arcas
Date: July, 14th 2010

Overview

In last decades there have been big technological changes which have caused a sharp break with existing technologies so far. One of the most important is Internet which has become the most important communications environment in history, with over one thousand millions of users worldwide.

Internet has undergone several changes in recent years. One of the new concepts that have arisen is Web 2.0. This philosophy is based on giving a more active role to the users, for example, using the collaboration of Internet users, also called collective intelligence; and the user-centered design.

This project describes the steps followed to develop a web 2.0 that provide a virtual world which emulates the Campus of the Baix Llobregat. It is described how the design and architecture is, and how the implementation to the different parts has done.

The result is called VirtualEPSC, a web 2.0 hosting a multimedia application in 2D that emulates the Campus of the Baix Llobregat and where users will be able to interact to each other. This platform allows promoting the campus in a funny way and people who want to know how the campus looks like can visit it in a virtual way. In addition, the application offers some services related to the campus and provides the possibility for registered users to create their own areas inside the virtual world, encouraging the participations of users.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

En sus orígenes, Internet tenía un objetivo específico: se navegaba con el propósito de buscar información. Este objetivo todavía prevalece, sin embargo, hoy en día es más probable perderse en la red debido al inmenso abanico de posibilidades que brinda, desde las redes sociales a la compartición de música, pasando por los juegos online, blogs, wikis, etc.

Todas estas nuevas posibilidades han surgido gracias a una serie de cambios que se han producido en Internet en los últimos años. Uno de los nuevos conceptos que han surgido es el de web 2.0. Este concepto admite diversas definiciones pero se podrían resumir en tres puntos:

- Segunda fase de Internet: web 2.0 es la transición que se ha dado de aplicaciones tradicionales hacia aplicaciones orientadas al usuario final. Es la segunda etapa de los proyectos y negocios de Internet, una vez superada la enorme crisis que se produjo a partir de 2000 con el estallido de la llamada “burbuja puntocom” [1].

- Web como plataforma: web 2.0 es una nueva manera de ofrecer servicios en Internet gracias a la combinación de diversas tecnologías que permiten utilizar la red como una plataforma de aplicaciones, lo que abre grandes posibilidades creativas.

- El usuario es el rey: web 2.0 es una etapa en la que el usuario adquiere un gran protagonismo (democratización de la web). Pasa de ser un mero espectador y consumidor de lo que le ofrece Internet, a convertirse en creador y generador de contenidos y servicios. Es un usuario que participa de manera activa, un superusuario.

Algunos ejemplos del uso de esta nueva filosofía 2.0 son blogs, wikis, redes sociales o plataformas educativas. Estas aplicaciones están triunfando en Internet, ya que ofrecen una amplia variedad de servicios y permiten a los usuarios interactuar entre ellos de diversas maneras.

Siguiendo esta línea, surge la idea de desarrollar una plataforma basada en la filosofía 2.0 y que esté relacionada con el ámbito universitario de la EPSC.

El objetivo de este trabajo es la creación de una web 2.0 que aloje un mundo virtual en 2D que permita promocionar la escuela de una manera divertida. Este mundo virtual debe simular el Campus del Baix Llobregat, ofrecer servicios relacionados con el campus y permitir a los usuarios interactuar unos con otros siguiendo el modelo de las redes sociales. Además, el sitio web ha de proporcionar la posibilidad de que usuarios registrados puedan crear sus propias zonas dentro del mundo virtual, incentivando aún más la participación de los usuarios.

El trabajo está divido en 6 capítulos que se estructuran de la siguiente manera:

En el primer capítulo, se hace una pequeña introducción al concepto de web 2.0 y cuáles son las más populares actualmente. En este punto, también se definen de manera específica cuáles son los objetivos del proyecto. El capítulo finaliza con una pequeña historia que describe el funcionamiento de la aplicación desde el punto de vista de un usuario.

El segundo capítulo describe cuáles son las especificaciones del proyecto. En este apartado, se proporciona una visión global del proyecto.

En el tercer capítulo, se describe cómo son el diseño y la arquitectura de la web y la aplicación del mundo virtual. Además, se especifica con qué tecnologías se han implementado las diferentes partes del trabajo.

En el cuarto capítulo, se mencionan el resto de las tecnologías utilizadas que han facilitado el desarrollo del proyecto.

El quinto capítulo muestra cuál ha sido el plan del proyecto, especificando las diferentes tareas y el tiempo de dedicación.

Finalmente, en el sexto capítulo, se encuentran las conclusiones del trabajo.

CAPÍTULO 1. MOTIVACIÓN Y OBJETIVOS

En este capítulo, se hace un estado del arte sobre la evolución de las webs y los juegos 2.0, seguidos de una descripción de los objetivos del trabajo. A continuación, se describe cómo debe ser una web 2.0 y se acaba con una pequeña historia que describe cuáles son las funcionalidades básicas de la aplicación desde el punto de vista de un usuario.

1.1. CONCEPTOS BÁSICOS

Actualmente, existe una tendencia a crear sitios webs y aplicaciones siguiendo la filosofía 2.0. Esta filosofía se basa en la colaboración de los usuarios en Internet, también llamado inteligencia colectiva, y en el diseño centrado en el usuario.

Por esta razón, surge la idea de crear una web 2.0 que alojará una aplicación multimedia en 2D, donde los usuarios podrán interactuar entre ellos. Esta aplicación se basa en un mundo virtual que emula el Campus del Baix Llobregat y donde los usuarios son representados mediante avatares.

Este proyecto permite promocionar el campus de una manera divertida, dando la posibilidad de que las personas que quieran conocerlo puedan visitarlo de una manera virtual y ver cuáles son los servicios que se ofrecen.

Además, la aplicación proporciona una serie de servicios a aquellas personas que ya son miembros de la escuela. Algunos ejemplos de éstos pueden ser accesos rápidos a Atenea, NetÁrea, Bibliotécnica, etc.

En este trabajo se han tomado como ejemplo otras aplicaciones 2.0 que ya están en marcha:

- Second Life [2]: este software es un universo virtual en 3D accesible por Internet y desarrollado por Linden Lab. Mediante un programa cliente, los usuarios pueden interactuar mediante sus avatares con el resto de miembros. Estos usuarios pueden explorar el mundo virtual, conocer gente, participar en eventos, crear sus propiedades o servicios públicos, etc. El atractivo de esta aplicación es que permite vivir una vida virtual paralela donde se pueden cumplir los sueños de los jugadores.

- UMH Virtual [3]: la Universidad Miguel Hernández de Alicante ha creado un software que emula sus campus universitarios de forma virtual en 3D. UMH Virtual es una página web, que mediante un plugin diseñado por ellos, permite crear avatares, caminar por los diferentes campus, hablar con otros usuarios y conocer la ubicación de los servicios que ofrecen.

1.2. OBJETIVOS DE DESARROLLO

El objetivo principal del proyecto es proporcionar una serie de servicios relacionados con el campus de una forma diferente a la habitual y conseguir integrar a los participantes en el desarrollo y la ampliación del proyecto.

Para conseguir esta meta, se han propuesto los siguientes objetivos específicos:

- Desarrollar una plataforma de un mundo virtual en 2D.
- Incentivar la participación de los usuarios.
- Democratizar el sistema, que todos los usuarios puedan crear nuevas zonas del mundo virtual.
- Ofrecer un servicio para que los miembros puedan interactuar entre ellos.
- Ofrecer un acceso rápido y sencillo a servicios relacionados con el campus.
- Incluir material lúdico y de entretenimiento.
- Ofrecer un servicio de noticias para mantener a los usuarios al corriente de las últimas actualizaciones.
- Facilitar el registro y la identificación de los usuarios utilizando perfiles o cuentas ya creadas.
- Utilización de un sistema de almacenamiento de datos escalable delante de grandes volúmenes de información.

Este proyecto se centra en la creación de una web 2.0 orientada a estudiantes, profesores y cualquier persona que quiera conocer el Campus del Baix Llobregat. La característica principal de la web es el mundo virtual, una aplicación donde los usuarios pueden visitar el campus de forma virtual.

En este campus virtual, los usuarios tienen la posibilidad de interactuar entre ellos mediante chats públicos, así como también utilizando chats privados entre solamente dos personas. Otros servicios que se ofrecen en esta aplicación son enlaces a páginas web de interés o servicios online del campus, también contiene enlaces a juegos relacionados con la universidad y con las diferentes zonas del campus.

La aplicación pretende ser una herramienta de soporte para los estudiantes y profesores, y que estos puedan acceder a los servicios del campus de forma rápida. Pero también sirve para promocionarlo de una manera diferente; poder visitar las diferentes zonas y conocer mejor todas sus características de forma virtual.

Por otro lado, la aplicación también cuenta con un apartado de red social, ya que los usuarios se pueden relacionar entre ellos mediante chats y los diferentes juegos. Para fomentar este uso social existe una competición entre los usuarios. Jugando a estos juegos del campus virtual se consiguen puntos sociales, de manera que los cinco mejores jugadores aparecen en una lista.

Además, el portal web cuenta con dos funciones más: el perfil de usuario y las noticias.

El perfil de usuario contiene información personal, como el nombre, el correo electrónico, las aficiones, etc., y también una imagen de usuario. La información y la imagen se pueden modificar en cualquier momento. Este perfil es público, ya que todos los usuarios registrados pueden consultar el perfil del resto.

Las noticias son añadidas por los administradores o desarrolladores de la web y las pueden leer todos los usuarios. Las noticias están relacionadas con la web y la aplicación, además pueden estar acompañadas de una fotografía. Éstas aparecen en orden cronológico, de la más reciente hasta la más antigua.

1.3. WEB 2.0

El concepto web 2.0 surge en una reunión, en el año 2005, entre dos empresas punteras del sector tecnológico: O'Relly Media y MediaLive. En esa reunión se analizaron diferentes empresas y se descubrió que lejos de hundirse, Internet estaba repleto de nuevas y fascinantes aplicaciones lanzadas por dichas empresas. A raíz de esta reunión, se creó la conferencia sobre web 2.0, web 2.0 SUMMIT [4], que se sigue celebrando anualmente.

El concepto 2.0 es muy amplio y no se puede definir sólo como una tecnología o conjunto de tecnologías, sino más bien como una filosofía de desarrollo y uso del software. Todo sitio web 2.0, debe cumplir alguna de las siguientes reglas:

- *La Web como plataforma*.
  Es importante cambiar la forma de ver la web. Ésta ha de servir de interfaz de usuario para todo tipo de aplicaciones, y no sólo para visualizar contenidos.

- *Aprovechar la inteligencia colectiva.*
  Se debe confiar en el usuario para que contribuya con sus propios contenidos a la aplicación o pueda revisar los existentes.

- *Sindicación*.
  Consiste en avisar a los usuarios cuando se produce algún cambio en la web. Esto es posible mediante la tecnología RSS basada en XML, que consiste en enviar al usuario la lista de nuevos contenidos.

- *Grandes cantidades de datos.*
  En las webs 2.0 es necesaria una base de datos altamente compleja y que disponga de facilidades para que los usuarios puedan añadir contenidos.

- *Beta perpetua.*
  Las webs 2.0 han de estar siempre en constante evolución, buscando nuevas funcionalidades, solucionando los problemas potenciales y agregando nuevos contenidos.

- *Diseño centrado en el usuario.*
  Con la aparición de nuevas tecnologías como AJAX y Flash, se busca que la apariencia de la página web sea como la de una aplicación de escritorio. Esto debe dar una facilidad y comodidad de uso al usuario: la información debe poderse introducir y extraer fácilmente, los usuarios deberían controlar su propia información, la existencia de enlaces es algo imprescindible, y el portal web debe poder ser utilizado enteramente desde un navegador. Además, cualquier sitio web 2.0 debe permitir que los usuarios interactúen unos con otros.

Para cumplir estas características, todo sitio web 2.0 debe utilizar algunas de estas técnicas:

- CSS, marcado XHTML válido semánticamente.
- Técnicas de aplicaciones ricas no intrusivas (como AJAX).
- Java Web Start.
- XUL.
- Redifusión o Agregación de datos en RSS/ATOM.
- URLs sencillas con significado semántico.
- JCC y APIs REST o XML.
- JSON.
- Algunos aspectos de redes sociales.
- Mashup (aplicación web híbrida).

Algunos ejemplos de web 2.0 pueden ser las wikis, los blogs, las redes sociales o las plataformas educativas.

En este trabajo se pretende trabajar con CSS, AJAX, REST, Mashup (como Google Maps) y conceptos de redes sociales.

1.4. DIARIO DE UN ESTUDIANTE

Empieza el nuevo curso académico y, de la misma manera, ocurre en el mundo virtual del Campus del Baix Llobregat. Un día cualquiera para Pepe, comienza con la llegada a la estación de Renfe de Castelldefels. A toda prisa, se dirige a su primera clase en el laboratorio 333, donde algunos de sus profesores de la optativa de IST (como viene siendo habitual) han preparado una práctica. Ésta consiste en un pequeño trivial con preguntas de telemática.

Como Pepe no es un experto en este tema, le han quedado algunas dudas sobre las respuestas, decide ir a consultas. Afortunadamente, ha podido resolver sus dudas mediante una conversación con su profesor, ya que éste se encontraba en el campus.

Llega la hora de la comida y Pepe junto algunos de sus compañeros, toman un apetitoso menú en el bar del campus. Como la vida de estudiante es muy dura, mientras toman un café deciden echar una timba de póker.

Pepe y sus amigos viven pendientes de una fecha en su calendario, el día de la Castelldefesta. Ese día está repleto de actividades, que pondrán a Pepe a prueba: carreras de sacos, concurso de birras, lanzamientos de huevos, etc. Cada vez que Pepe participa en juego del campus, aumenta su cantidad de puntos sociales. A medida que aumente su puntuación podrá entrar en el top5 de los cinco mejores participantes.

En una hora de descanso, como Pepe está muy estresado después de su clase, se va a dar un paseo y navegar en el lago del campus virtual.

Más tarde, decide crear su propia zona en el mundo virtual para poder hablar con sus amigos. Para elegir la zona busca un mundo vacío en el mapa. La intención de Pepe es crear un campo de fútbol y una fuente para su mundo virtual. Para ello, crea los archivos necesarios que están especificados en la web y compra el mundo desde el mapa de la aplicación.

CAPÍTULO 2. ESPECIFICACIONES

Este capítulo describe cuáles son las especificaciones generales del proyecto. Básicamente, se divide en dos grandes bloques: el sitio web y la aplicación del campus virtual.

2.1. SITIO WEB

El sitio web se define como un conjunto de páginas web que permiten al cliente interactuar con el portal mediante peticiones HTTP. Concretamente, el portal web ha sido diseñado basándose en el concepto de web 2.0.

2.1.1. Tipos de Usuarios

Una vez registrado en la web al usuario se le asigna un rol por defecto. Sin embargo, existen tres tipos de roles:

- *Usuario*: es asignado a todos aquellos usuarios que únicamente tienen intención de hacer uso de la aplicación. Además, este rol permite al usuario modificar su perfil, jugar, ver las últimas noticias publicadas y visitar perfiles de otros usuarios, entre otras cosas.

- *Desarrollador*: goza de todas las funcionalidades de usuario, pero a diferencia de un usuario corriente, el rol de desarrollador permite editar el sitio web y gestionar a los usuarios con rol de usuario o desarrollador.

- *Administrador*: es el rol que goza con más permisos. Permite la gestión de usuarios con cualquier tipo de rol y editar la web.

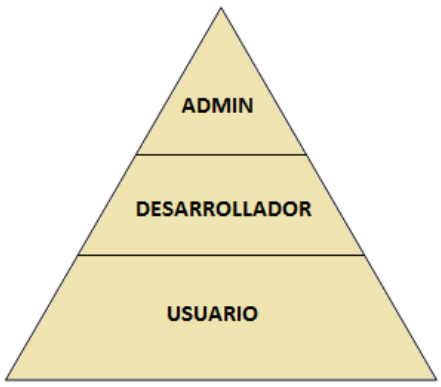

Fig. 1 Pirámide de la jerarquía de usuarios

2.1.2. Funcionalidades

La tabla 1 muestra cuáles son las diferentes funcionalidades que se ofrecen en la web y para que tipos de usuarios están disponibles.

Tabla 1 Funcionalidades de la web

FUNCIONALIDADES	TIPOS DE USUARIOS (ROLES)		
	Usuario	***Desarrollador***	***Administrador***
Editar perfil propio de usuario *(nombre, apellidos, foto, etc.)*	✓	✓	✓
Edición mínima de roles de usuarios *(cambio de usuario a desarrollador)*		✓	✓
Edición máxima de roles de usuarios *(cambio de usuario a desarrollador o administrador)*			✓
Eliminación de usuarios *(con rol de usuario o desarrollador)*		✓	✓
Eliminación de usuarios *(con cualquier tipo de rol)*			✓
Enviar incidencias mediante el formulario de contacto	✓	✓	✓
Atender incidencias o dudas de la herramienta web			✓
Acceso a la aplicación del Mundo Virtual	✓	✓	✓
Edición del portal web *(agregar noticias, etc.)*		✓	✓
Buscar y visitar perfiles de otros usuarios	✓	✓	✓
Agregar nuevas zonas al mundo virtual	✓	✓	✓
Acceso a las noticias	✓	✓	✓

2.1.2.1. Árbol de contenidos

Para ilustrar la estructura del sitio web se hace uso de un árbol de contenidos. En el esquema de la figura 2, se pueden ver cuatro tipos de recuadros de colores. Los recuadros azules hacen referencia a páginas web que son accesibles a cualquier visitante de la web. Los recuadros de color verde

únicamente son accesibles a aquellos usuarios que hayan pasado por el proceso de autenticación correctamente. Por otro lado, los recuadros de color violeta representan las páginas web que únicamente son accesibles a aquellos usuarios que gozan de un rol de administrador o desarrollador.

Finalmente, en este árbol de contenidos también se representa la ubicación de la aplicación del mundo virtual dentro del sitio web.

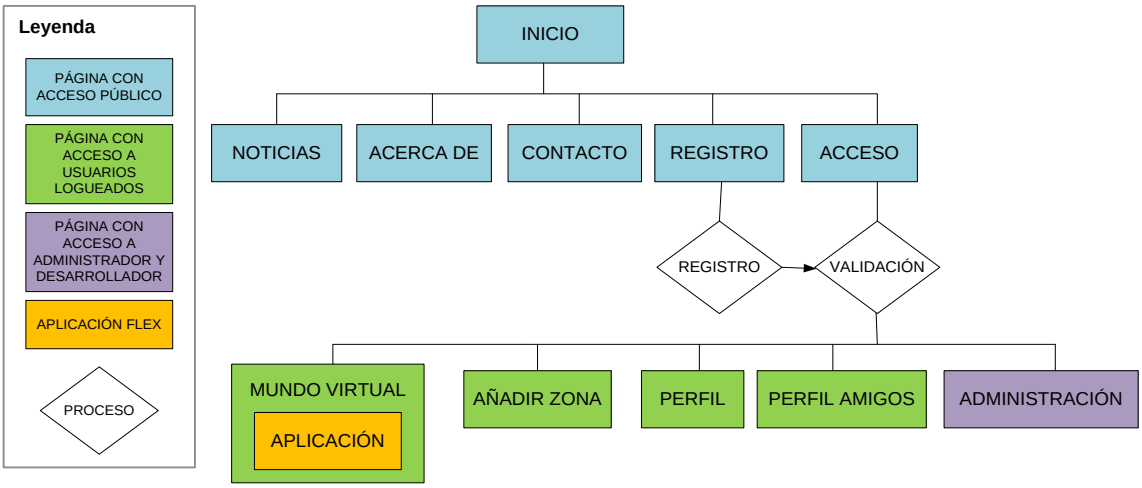

Fig. 2 Esquema de la web

2.1.2.2. Registro y perfil de usuario

Para poder utilizar todas las funcionalidades de las que se dispone en la web es necesario realizar un registro. Para ello, se ha habilitado un formulario de registro típico donde se exigen unos datos mínimos de información sobre el usuario.

Actualmente, este proceso de registro mediante formulario suele ser el primer escollo que se debe superar para que los usuarios utilicen un servicio. Por esta razón, también se permite realizar el registro en la web mediante una cuenta OpenID existente (véase apartado 3.4). Con tan sólo un clic de ratón y confirmando que sus datos son correctos, el usuario realiza el registro en la web.

Una vez registrado en el portal, se crea un perfil de usuario donde se pueden modificar los datos principales, como aficiones, contraseña, etc. Así como la foto que se muestra en el perfil de usuario.

De la misma manera que el usuario es capaz de ver su propio perfil, éste puede visualizar otros perfiles de usuarios que han sido previamente registrados.

2.1.2.3. Servicio de noticias

Además, la web cuenta con un servicio de noticias, mediante el cual se pretende mantener informados al resto de usuarios de la web sobre las últimas novedades.

Las noticias pueden ir desde la ampliación de zonas disponibles para visitar en el mundo virtual hasta nuevas funcionalidades en la web. Además, estas noticias pueden tener imágenes adjuntas para reforzar la información.

Todas las noticias se muestran ordenadas por la fecha de edición. De esta manera, las últimas noticias se encuentran en primera plana, y en las páginas siguientes, se encuentran noticias anteriormente añadidas.

Como ya hemos mencionado, únicamente pueden agregar noticias aquellos usuarios que dispongan de un rol de administrador o desarrollador.

2.2. APLICACIÓN DEL MUNDO VIRTUAL

El objetivo principal del portal web es dar soporte a una aplicación 2D que emula un mundo virtual del Campus del Baix Llobregat. Todos los usuarios que estén registrados en la web pueden participar en el campus virtual mediante un avatar que lo representa y ver al resto de usuarios conectados.

La tabla 2 muestra cuáles son las diferentes funcionalidades que se ofrecen en la aplicación del mundo virtual.

Tabla 2 Funcionalidades de la aplicación

Funcionalidades de la aplicación
Pasear por las diferentes zonas virtuales del campus.
Elegir un avatar.
Consulta de un miniperfil de juego.
Consulta del Top5 (5 jugadores con más puntos sociales).
Consulta del miniperfil del resto de usuarios.
Añadir a otros usuarios como amigos.
Ver los amigos conectados.
Consultar de forma fácil el miniperfil de los amigos conectados.
Chat público en cada una de las zonas.
Chats privados entre dos usuarios conectados.
Información sobre novedades de los amigos.

Links a servicios online del campus u otras webs.
Links a juegos, con la posibilidad de incrementar los puntos sociales.
Consulta del mapa general del campus virtual.

2.2.1. Zonas del mundo virtual

El mundo virtual se divide en zonas independientes, que pueden ser exteriores o interiores, y representan los diferentes lugares del campus, como por ejemplo la biblioteca, la cafetería, la sala de actos, etc. Por ejemplo, está la entrada a la EPSC que es una zona exterior, y la sala de actos que es una zona interior, ya que está dentro de la zona de la entrada a la EPSC.

Por otra parte, dentro de cada una de las zonas del juego se pueden encontrar diferentes servicios: un chat público para cada zona (véase apartado 2.2.6), vínculos a juegos o a servicios online del campus.

Por ejemplo, en la zona de la biblioteca hay la posibilidad de acceder a la página web de bibliotécnica de la UPC. De la misma manera, se pueden disfrutar de entretenidos juegos, como el ping-pong que se encuentra en la cafetería.

2.2.1.1. Mapa general

Cada una de las zonas tiene asignado un espacio dentro del mapa principal, que está definido por un fichero XML llamado maps.xml (véase anexo A). El mapa principal es una tabla de N columnas y M filas, estos dos valores se definen en el XML del mapa tal y como se muestra en la figura 3.

```
<map>
        <tilesX>4</tilesX> → Número de columnas
        <tilesY>3</tilesY> → Número de filas
</map>
```

Fig. 3 Ejemplo de definición de columnas y filas del mapa general

La siguiente figura muestra como es la vista del mapa general, donde se pueden observar cuales son todos los mundos exteriores y que zonas están vacías. Además, se puede apreciar como es la vista del interior de la zona del edificio de la biblioteca, donde se pueden ver tres zonas interiores y una vacía.

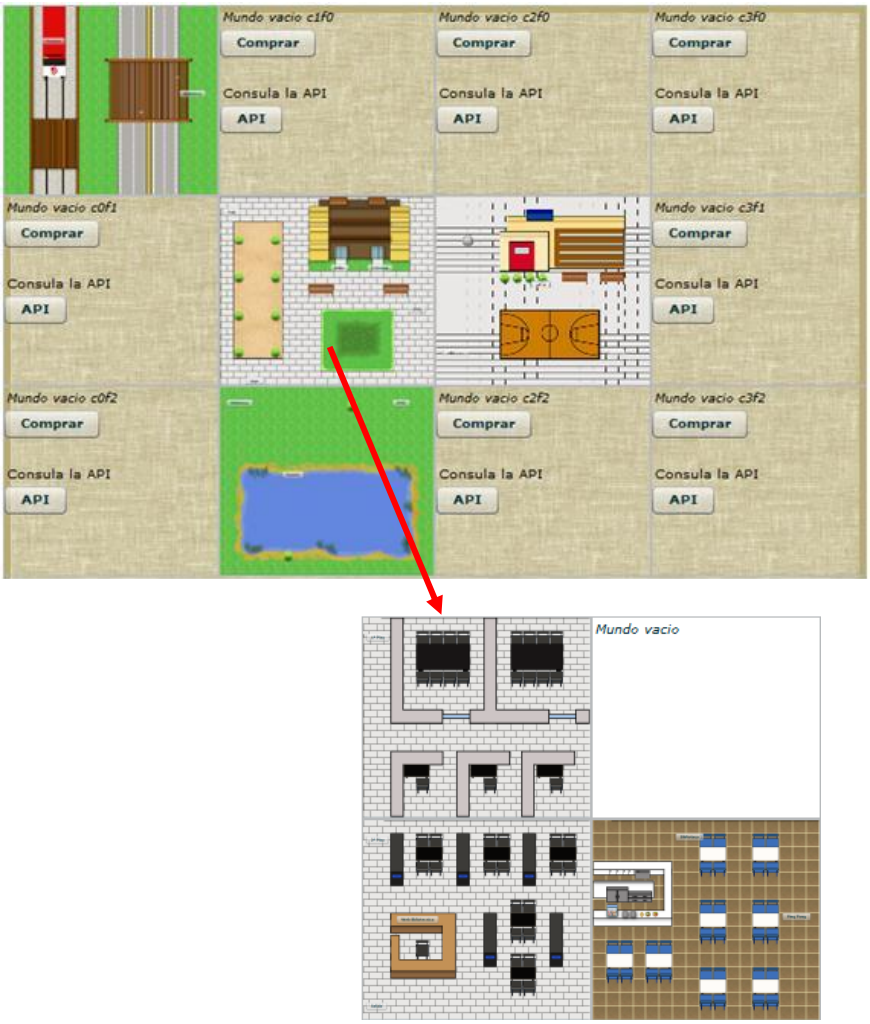

Fig. 4 Vista del mapa general y una de las vistas interiores

Para definir cada una de las zonas exteriores creadas en el campus, hay que añadir un nuevo "mundo" en el fichero del mapa general, tal y como muestra la figura 5. Dentro del "mundo" se indica en qué fila y qué columna de la tabla se sitúa. Como ya se ha mencionado anteriormente, dentro de las zonas exteriores puede haber más zonas interiores, por lo tanto, también hay que indicar como es la tabla interior de ese mundo, es decir, el número de columnas y filas.

```
<world>
        <tileX>1</tileX> → Posición del mundo en el mapa general (columna)
        <tileY>2</tileY> → Posición del mundo en el mapa general (fila)
        <name>Lago</name> → Nombre descriptivo de la zona
        <source>logo-c1f2.png</source> → Logo
        <tilesX>2</tilesX> → Número de columnas interiores
        <tilesY>2</tilesY> → Número de filas interiores
</world>
```

Fig. 5 Ejemplo de definición de una zona exterior

De la misma manera, para definir una zona interior, hay que añadir un nuevo "mundo" interior dentro de la definición de la zona exterior a la que pertenece, como se ve en la figura 6. Dentro del "mundo" interior se indica en qué fila y columna de la tabla se sitúa.

```
<worldI>
      <tileX>0</tileX> → Posición del mundo en la tabla interior (columna)
      <tileY>1</tileY> → Posición del mundo en la tabla interior (fila)
      <name>Lab 333</name> → Nombre descriptivo de la zona
      <source> logo-c2f1ic0f1.png</source> → Logo
</worldI>
```

Fig. 6 Ejemplo de definición de una zona interior

Las zonas de la tabla del mundo general que no estén definidas, simplemente aparecen como "mundos" que se encuentran todavía vacios. Lo mismo ocurre con las zonas que no están definidas en las tablas interiores.

Como es lógico, dos zonas diferentes no pueden colocarse en la misma posición de la tabla. Por esta razón, es necesario diferenciar las diferentes zonas según su situación en el mapa. Para ello, se ha establecido una regla que indica que para una zona exterior, su nombre seria CNFM, donde N es la columna y M la fila donde se encuentra en el mapa general. Para una zona interior, el nombre sería de la siguiente manera CNFMICXFY, donde N y M son la situación de la zona exterior al que pertenece, X la columna e Y la fila en las que se sitúa dentro de la tabla interior. La figura 7 muestra un ejemplo de cómo se aplica esta regla.

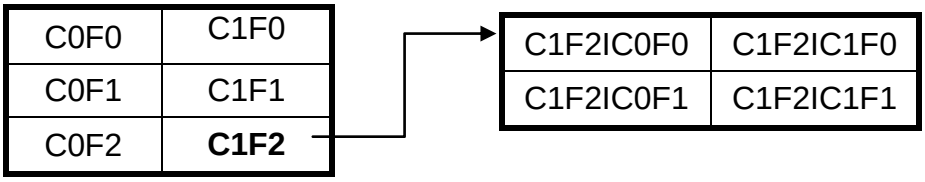

Fig. 7 Nombres de las zonas según su posición

2.2.1.2. Zonas individuales

Cada una de las zonas está definida por un fichero XML. El nombre de los ficheros se define con la regla explicada anteriormente. Por ejemplo, la zona que está en la columna 0 y fila 0, tendría un fichero llamado worldc0f0.xml. En estos ficheros se describen las zonas, cómo son visualmente, su nombre, etc.

Las características principales que se configuran son:

- *El nombre de la zona*: el nombre indica en que zona se encuentra el usuario, como pueden ser la estación de tren o la 1º planta de la biblioteca. Esta propiedad se modifica tal y como muestra la figura 8.

```
<name>entrada biblioteca</name>
```

Fig. 8 Ejemplo de definición del nombre de una zona

- *La vista de la zona*: especifica donde se sitúan los diferentes dibujos en el mapa. Se divide en tres partes, cada una de ellas con sus respectivos objetos:
    - *El nivel de suelo*: indica que dibujo hay en el suelo. Se define con el nombre del dibujo que se va a usar, y la longitud y altura de éste. Se puede ver un ejemplo en la siguiente figura.

```
<ground>
   <source>losa</source>
   <width>40</width>
   <height>40</height>
</ground>
```

Fig. 9 Ejemplo de definición del nivel de suelo de una zona

En la figura 10, se muestra como es el nivel de suelo de una de las zonas, concretamente de la entrada de la EPSC.

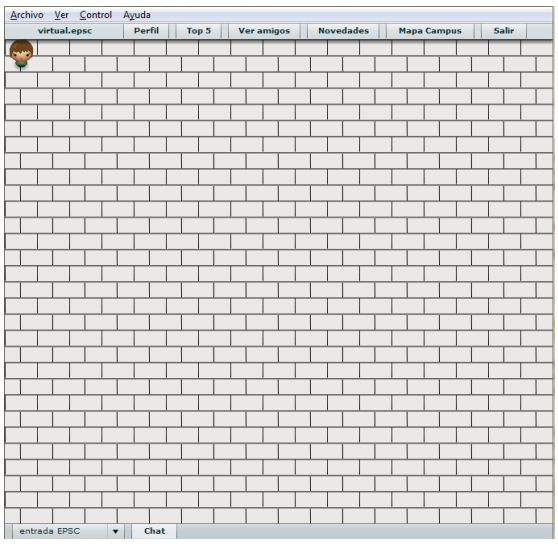

Fig. 10 Vista del nivel de suelo

- *El nivel de baldosas*: son aquellos objetos que están encima del suelo pero que los avatares quedan dibujados por encima. Estos objetos pueden tener colisión.

```
<baldosas>
   [...]
</baldosas>
```

Fig. 11 Ejemplo de definición del nivel de baldosas de una zona

La figura 12 muestra como es la zona de la entrada a la EPSC únicamente con el nivel de suelo y el nivel de baldosas, se puede ver como el avatar queda dibujado encima de los nuevos objetos (la pista de básquet y los bancos).

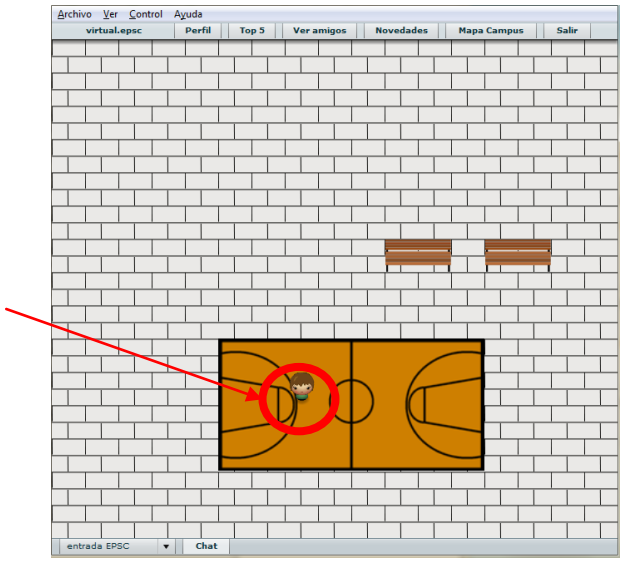

Fig. 12 Vista del nivel de baldosas

- *El nivel de buildings*: son aquellos objetos que están encima del suelo pero que se dibujan por encima de los avatares. Estos objetos también pueden tener colisión.

```
<buildings>
   [...]
</buildings>
```

Fig. 13 Ejemplo de definición del nivel de buildings de una zona

En la figura 14, se puede ver la zona de la entrada a la EPSC con los tres niveles completos: suelo, baldosas y buildings. En este caso, se muestra como el avatar queda dibujado por detrás de los nuevos objetos (la roca, los arbustos y el edificio de la EPSC).

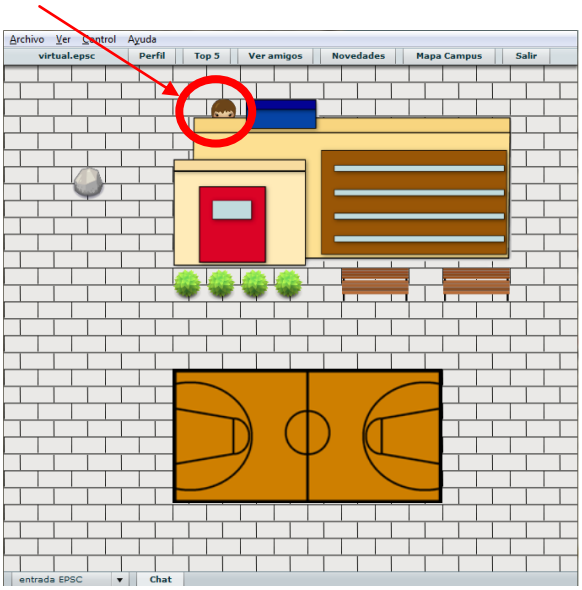

Fig. 14 Vista del nivel de buildings

Cada objeto del nivel de baldosas y el nivel de buildings se define por la columna y la fila donde se sitúan dentro del mundo, cuál es el objeto (es decir el nombre del dibujo que se va a colocar); y por la altura y la longitud del dibujo. Estos objetos pueden tener colisión, que significa que es una zona que los avatares no pueden pisar o entrar. La figura 15 muestra un ejemplo de la definición de un objeto de este tipo.

```
<build>
    <tileX>3</tileX>
    <tileY>2</tileY>
    <source>pizarra</source>
    <width>200</width>
    <height>120</height>
    <colision>
          <tileX1>3</tileX1>
          <tileY1>2</tileY1>
           <tileX2>7</tileX2>
          <tileY2>4</tileY2>
    </colision>
</build>
```

Fig. 15 Ejemplo de definición del un objeto

- *Los botones para dirigirse a otras zonas del campus*: estos botones sirven para dirigirse a otras zonas del campus de forma rápida. Se definen tal y como muestra la figura 16.

```
<buttons>
  <button>
        <id>Tren</id>
        <next>c0f0</next>
        <posX>10</posX>
        <posY>10</posY>
  </button>
</buttons>
```

Fig. 16 Ejemplo de definición de un botón

- *Los botones para abrir links a páginas webs*: pueden haber más de un link o ninguno en cada zona. Estos botones se definen tal y como muestra la siguiente figura.

```
<webs>
  <web>
    <id>Web EPSC</id>
    <link>http://epsc.upc.edu/ca/</link>
    <posX>320</posX>
    <posY>10</posY>
  </web>
</webs>
```

Fig. 17 Ejemplo de definición de un link a web

- *Los botones para abrir juegos*: pueden haber más de un juego o ninguno en cada zona. La siguiente figura muestra cómo se define un botón de juego.

```
<games>
  <game>
    <id>Trivial</id>
    <link>http://dxat.ath.cx:8080/VirtualEPSC
/flash/Trivial.swf</link>
    <posX>200</posX>
    <posY>200</posY>
  </game>
</games>
```

Fig. 18 Ejemplo de definición de un botón de juego

2.2.2. Avatares

En el mundo virtual, un usuario interactúa con el resto de jugadores mediante un avatar que lo representa.

La primera vez que se accede a la aplicación se elige este avatar utilizando la interfaz de la figura 19. Los botones *Anterior* y *Siguiente* permiten visualizar los diferentes avatares existentes, y para escoger el preferido, simplemente hay que clicar en el botón *Aceptar*.

Fig. 19 Interfaz de elección de avatar

Si en cualquier momento se quiere escoger otro avatar, se puede modificar desde el miniperfil de juego, que se explicará en el apartado 2.2.4. En el momento en que se desea cambiar de avatar se vuelve a mostrar la misma interfaz de la figura 19.

2.2.3. Red de amigos

La aplicación contiene un apartado de red social, mediante la cual pueden establecerse amistades entre los usuarios.

Un usuario puede añadir como amigo a otros usuarios registrados que utilicen la aplicación. El concepto de amigo en el juego indica que interesa saber si ese usuario está conectado o no de forma rápida, lo que está haciendo en cada momento, establecer chats privados sin tener que buscar al jugador por las diferentes zonas, etc.

Para ello, existe una lista de amigos conectados, que permite, además de saber quién hay conectado, acceder a sus perfiles de forma rápida para poder saber su información o entablar chats privados.

2.2.3.1. Novedades

Otra funcionalidad de la aplicación es el menú de novedades donde van apareciendo los eventos principales de los amigos: quién se conecta o desconecta, si un usuario abre algún juego o pasea en el lago, etc.

2.2.4. Miniperfil de usuario

La aplicación tiene un miniperfil de usuario donde se puede consultar el nombre de usuario, el nombre completo, los puntos sociales y ver cuál es el avatar escogido de cualquier jugador.

Desde la interfaz de juego se puede acceder al miniperfil de usuario utilizando el botón *Perfil*. Además, existe la posibilidad de cambiar de avatar.

La figura 20 muestra el aspecto del miniperfil de usuario que se puede abrir desde la aplicación del mundo virtual.

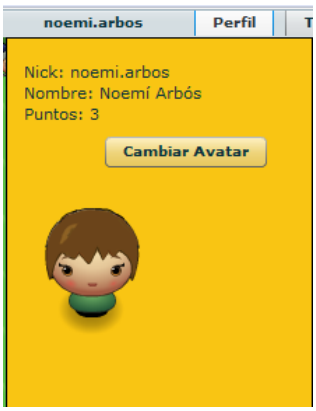

Fig. 20 Miniperfil de usuario

También es posible consultar el miniperfil del resto de usuarios que estén conectados. Para ello existen dos posibilidades:

- Desde el avatar del jugador: dentro de la misma zona, se puede clicar en el avatar de cualquier usuario y se abrirá una nueva ventana con sus datos.

- Desde la lista de amigos conectados: utilizando el botón *Ver Perfil* situado debajo del nombre de las amistades, se puede acceder de forma rápida a su miniperfil.

En el miniperfil del resto de usuarios no existe la posibilidad de cambiar el avatar, en cambio, hay dos funciones nuevas: añadir a ese usuario a la lista de amigos y abrir un chat privado con él. El botón de agregar como amigo sólo aparece si ese usuario todavía no está agregado en la lista de amistades.

En la figura 21 se pueden observar dos ejemplos de miniperfil de otros usuarios, el de la izquierda pertenece a un usuario que no ha sido agregado como amigo y el de la izquierda pertenece a un usuario que sí ha sido agregado.

Fig. 21 Miniperfil del resto de usuarios

2.2.5. Puntos Sociales

Cada vez que un usuario juega a alguno de los juegos que están distribuidos por diferentes zonas del campus, este usuario consigue puntos **"**sociales**"**.

Desde la aplicación se puede consultar la lista de los 5 usuarios con mayor puntuación. En la siguiente figura, se muestra como es el aspecto de esta lista de usuarios.

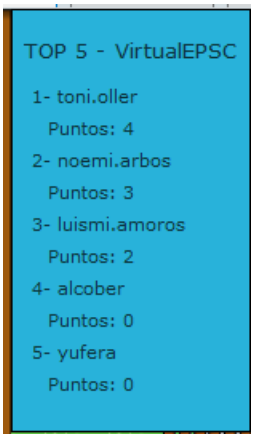

Fig. 22 Lista de los cinco mejores usuarios

2.2.6. Servicio de mensajería instantánea

La aplicación del mundo virtual cuenta con un servicio de chat, en el que todos los usuarios que se encuentren dentro de la misma zona se pueden comunicar. En este chat, todos los mensajes son públicos y cualquier usuario que se encuentre en esa zona del mundo virtual puede ver los mensajes del resto de jugadores. La figura 23 muestra como es el aspecto de uno de estos chats públicos.

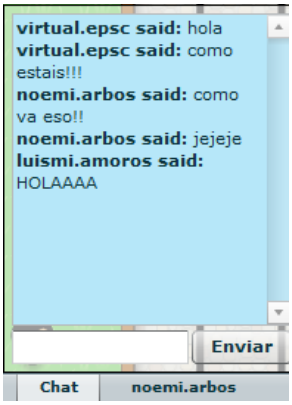

Fig. 23 Chat público de zona

Además del chat público, la aplicación cuenta con un chat privado entre usuarios, que se puede observar en la figura 24. Si un usuario desea contactar con otro usuario o amigo que está utilizando la aplicación, puede hacerlo mediante este chat privado independientemente de si se encuentra en la misma zona que él. Para abrir este chat se debe indicar desde el miniperfil del otro usuario, por lo tanto tiene que encontrarse en la misma zona del mundo virtual o estar en la lista de amistades para poder ver su miniperfil.

Fig. 24 Chat privado

CAPÍTULO 3. DISEÑO Y ARQUITECTURA

En este capítulo se presenta la definición arquitectónica y el diseño del proyecto. Este apartado se divide en diferentes partes, cada una de ellas corresponde con un bloque de la arquitectura general del proyecto, y se especifica cómo se ha realizado la implementación en cada bloque.

3.1. ARQUITECTURA GENERAL

La arquitectura general del proyecto puede ser algo compleja. Para entenderla de una manera más sencilla, se ha decidido realizar el siguiente esquema:

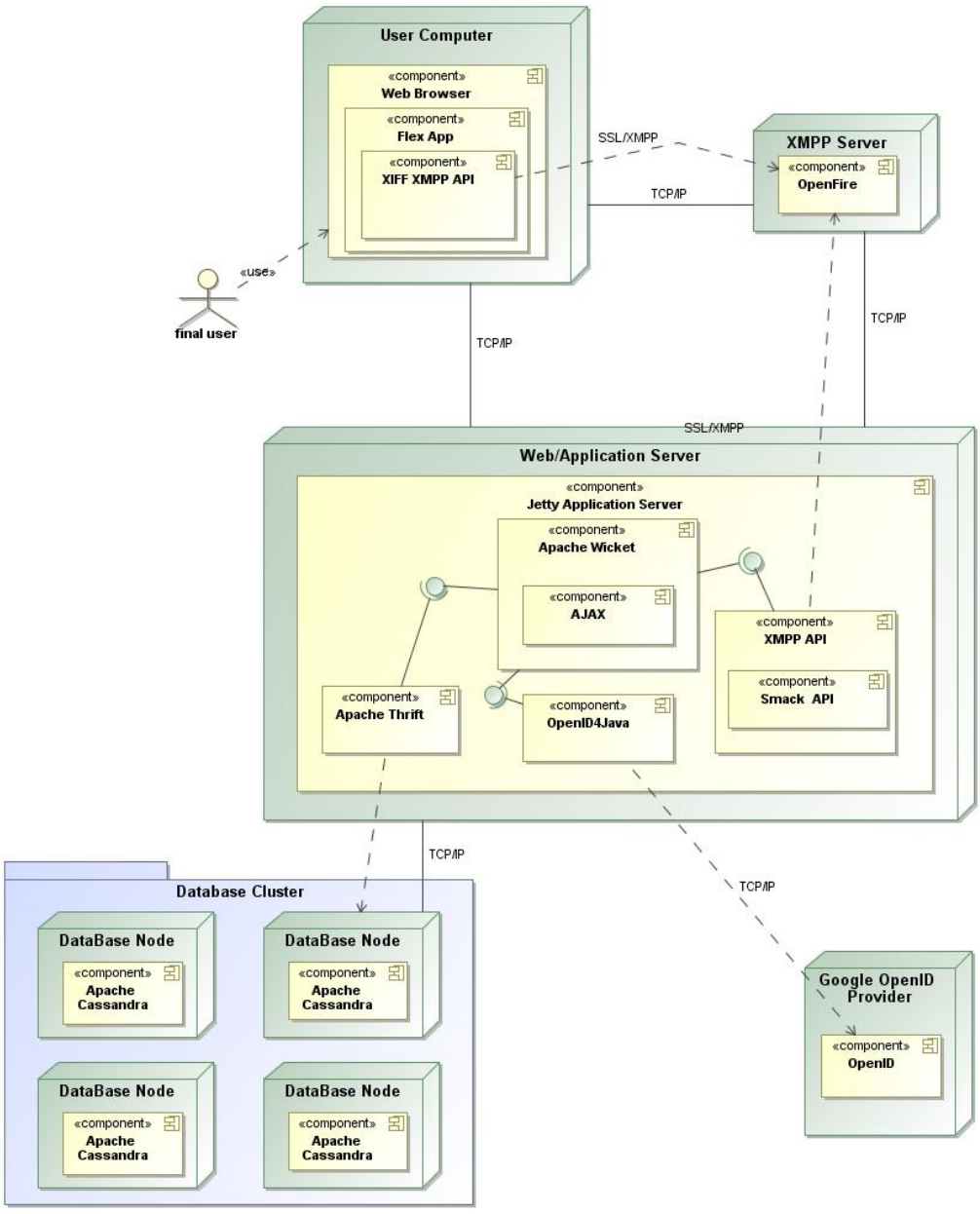

Fig. 25 Arquitectura del sistema

En este esquema se pueden observar cinco bloques principales:

- **Web/Application Server:** este bloque se encarga de ofrecer el portal web con todas sus funcionalidades (véase apartado 3.2) y de alojar la aplicación del mundo virtual.

  Para implementar el servidor web se ha utilizado Apache Wicket, la tecnología encargada de ofrecer la vista de las páginas web. Wicket además proporciona componentes de AJAX para utilizar en las páginas web.

- **Database Cluster:** el sistema de almacenamiento de datos (véase apartado 3.3) utiliza una de las últimas tecnologías de Apache llamada Cassandra, que permite una mayor eficiencia en tareas de lectura y escritura, además de ser mucho más robusta que otros sistemas. En este apartado también se incluye Apache Thrift, que permite la interacción entre el servidor web y el sistema de almacenamiento de datos.

- **Google OpenID Provider:** el proveedor de cuentas OpenID (véase apartado 3.4) se trata del servidor externo Google, a través de su API (GoogleOpenID) es posible implementar un registro y una autenticación mediante cuentas de Google, que consigue facilitar este proceso al usuario. Para conseguirlo este objetivo, se ha utilizado OpenID4Java en el servidor web.

- **XMPP Server:** es el encargado de gestionar la mensajería instantánea (véase apartado 3.5). Se ha decido utilizar el servidor Openfire para realizar esta función. En este proyecto la mensajería instantánea tiene dos funciones: la comunicación entre usuarios y el envío de notificaciones de movimientos o eventos de la aplicación del mundo virtual.

  En este bloque se incluye el componente Smack del servidor web, que es la tecnología encargada de interactuar con el servidor de mensajería instantánea en tareas de administración (creación de cuentas, eliminación de cuentas, etc). Además, se incluye la API de XIFF, un componente de la aplicación Flex encargado de proporcionar un cliente XMPP.

- **User Computer:** es, finalmente, en el navegador web del usuario donde se ejecuta la aplicación del mundo virtual y permite al usuario interactuar con el portal web. La aplicación ha sido programada utilizando Adobe Flex (véase apartado 3.6).

A continuación, se detallarán más a fondo cada una de las tecnologías que se han utilizado en el proyecto.

3.2. SERVIDOR WEB

La implementación de la vista del portal web se ha desarrollado utilizando Apache Wicket [5], conocido comúnmente como Wicket. Esta tecnología es uno de los últimos frameworks web de Apache para Java. Conceptualmente, es similar a otros frameworks web más conocidos como Java Server Faces, Tapestry o Struts, pero con mucha más proyección de futuro que éstos.

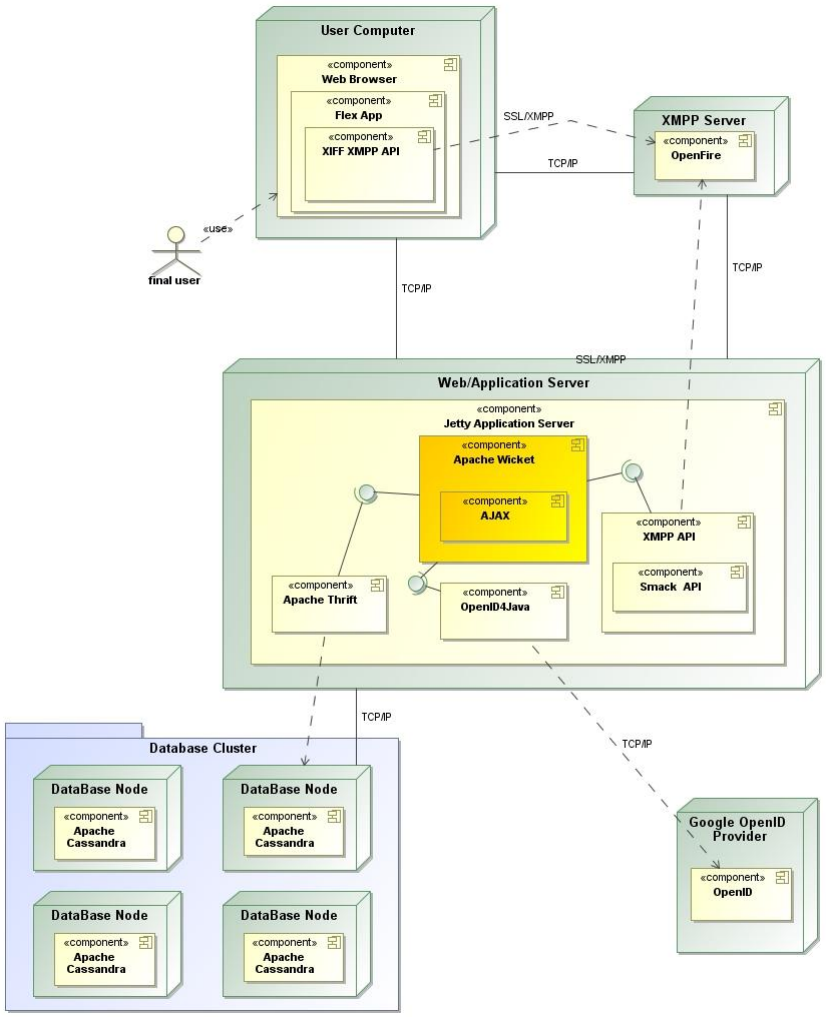

Fig. 26 Arquitectura del sistema – Servidor web

El equipo de Apache Wicket comenzó a desarrollar el framework teniendo dos premisas en la cabeza: separar claramente vista y lógica, y programar en Java todo el comportamiento [6].

Esto supone un cambio bastante importante a la hora de desarrollar aplicaciones web respecto a JSP, Struts o incluso JSF. En todos estos frameworks se confunden tags de vista con tags de lógica. Sin embargo, la idea de Wicket es coger el HTML, añadir un atributo wicket:id a cada tag que deba tener comportamiento dinámico, y gestionar toda la lógica de la página en la clase Java correspondiente. Para obtener más información sobre Apache Wicket, consultar el anexo B.

3.2.1. Implementación

En cuanto a la implementación del portal web, todas las páginas del portal comparten una serie de componentes. Para no tener que replicar el mismo código en todas las páginas, se ha decidido utilizar uno de los puntos fuertes de Wicket, la herencia de markups.

En primer lugar, se ha diseñado la página BasePage de la cual extenderán el resto de páginas. Esta BasePage está formada por 6 paneles:

- *Panel de cabecera:* se encarga de mostrar el titulo de la web junto con la imagen del campus. Además, contiene el menú de opciones del portal donde algunas de estas opciones son invisibles hasta que el usuario no está logueado, como se puede observar en la figura 27.

Fig. 27 Panel de cabecera

- *Panel de acceso:* se encarga de obtener la información para realizar el acceso del usuario. Una vez el usuario ha accedido a la web correctamente, este panel muestra información personal del usuario.

  La figura 28, muestra el panel de acceso en el momento anterior a la autenticación de usuario. También muestra, el panel que aparece una vez el usuario se ha logueado satisfactoriamente. Este panel muestra información útil del usuario: nombre de usuario, nombre completo y fecha de conexión.

Fig. 28 Panel de acceso

- *Panel de noticias:* su tarea es mostrar un resumen actualizado de las dos últimas noticias añadidas en la web, cada noticia consta de su título, fecha de edición y contenido. En la figura 29, se muestra un ejemplo del panel de noticias.

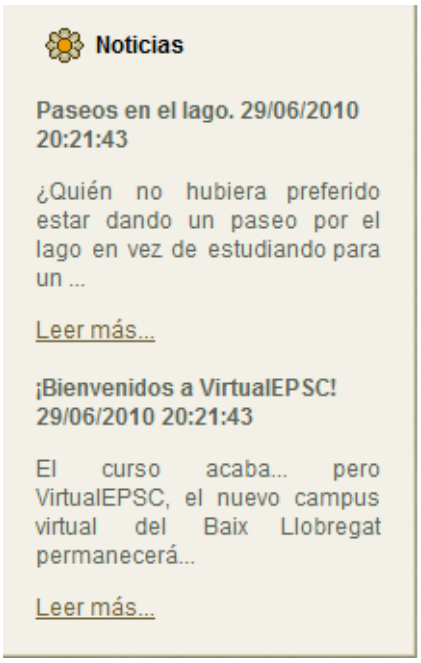

Fig. 29 Panel de noticias

- *Panel de búsqueda de usuarios:* su función es realizar búsquedas de usuarios registrados de manera dinámica mediante AJAX, donde según se van introduciendo caracteres sugiere posibles usuarios, como se muestra en la figura 30.

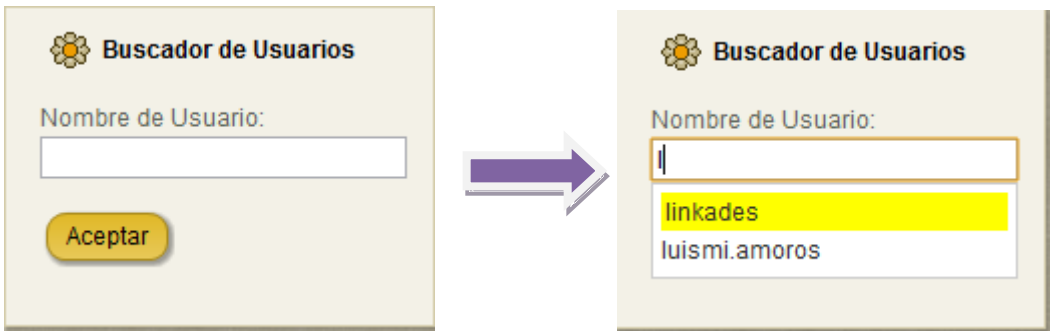

Fig. 30 Panel de búsqueda de usuarios

- *Panel de contacto:* muestra información de interés sobre el Campus del Baix Llobregat y los autores. Este panel, tal y como se puede observar en la figura 31, consta de un mapa mostrando la situación del campus y sus vías de acceso. Además muestra la dirección y el teléfono de contacto de la EPSC, así como también, proporciona unos emails de contacto de los creadores.

Fig. 31 Panel de contacto

- *Panel de pie de página:* se encarga de mostrar información sobre los autores y licencias del producto. Un ejemplo de este panel puede ser el de la figura 32.

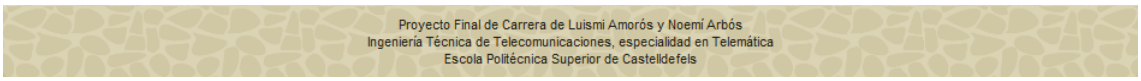

Fig. 32 Panel de pie de página

Por tanto, el resultado de esta BasePage que incluye los paneles mencionados anteriormente, tiene el aspecto mostrado en la figura 33:

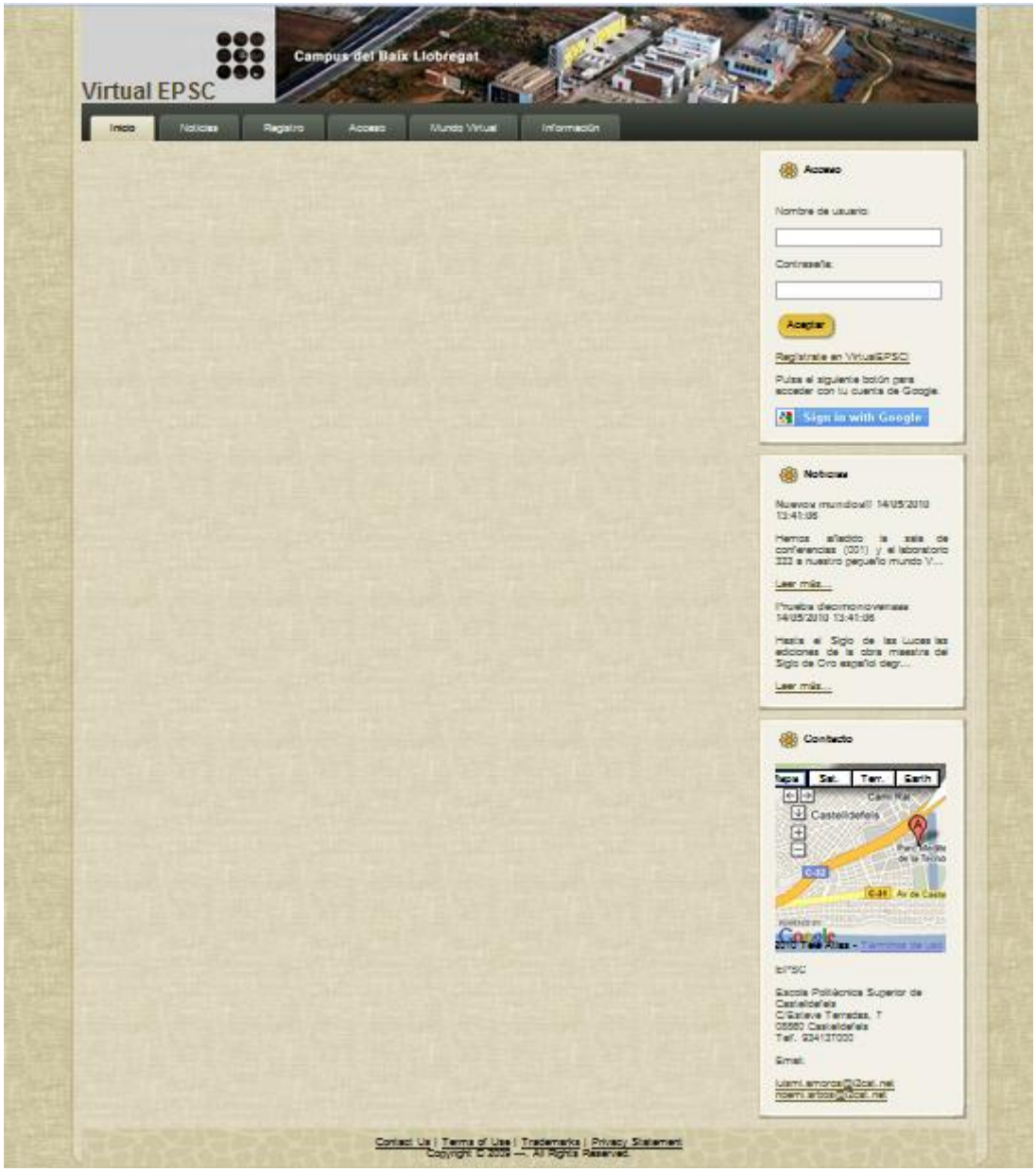

Fig. 33 Vista de la BasePage

Una vez diseñada la página que servirá de base para el desarrollo del resto del portal web, en primer lugar, cabe destacar que existen tres tipos de páginas:

- *Páginas visibles para cualquier visitante:* se tratan de todas aquellas páginas que pueden ver aquellos usuarios que entren en la web y no estén logueados. Un buen ejemplo de estas páginas pueden ser la de Inicio, Registro o Acceso.

- *Páginas visibles únicamente a usuarios logueados:* son aquellas páginas que pueden ser visitadas únicamente si el usuario se ha logueado previamente de manera correcta. Un ejemplo de este tipo es la página de Mundo Virtual.

- *Páginas visibles para usuarios específicos:* este tipo de páginas son sólo visibles a un determinado grupo de usuarios registrados en la web. Como ya se ha mencionado anteriormente, existen tres tipos diferentes de usuarios (véase apartado 2.1.1): Usuario, Desarrollador y Administrador. Este tipo de páginas solo están disponibles para usuarios del tipo Desarrollador o Administrador. Un ejemplo de ellas es la página de Admin.

A continuación, se muestra un diagrama de componentes del portal web de VirtualEPSC. Con este esquema se pretende aclarar cómo está formada la capa de presentación de la web siguiendo el esquema MVC (Modelo Vista Controlador).

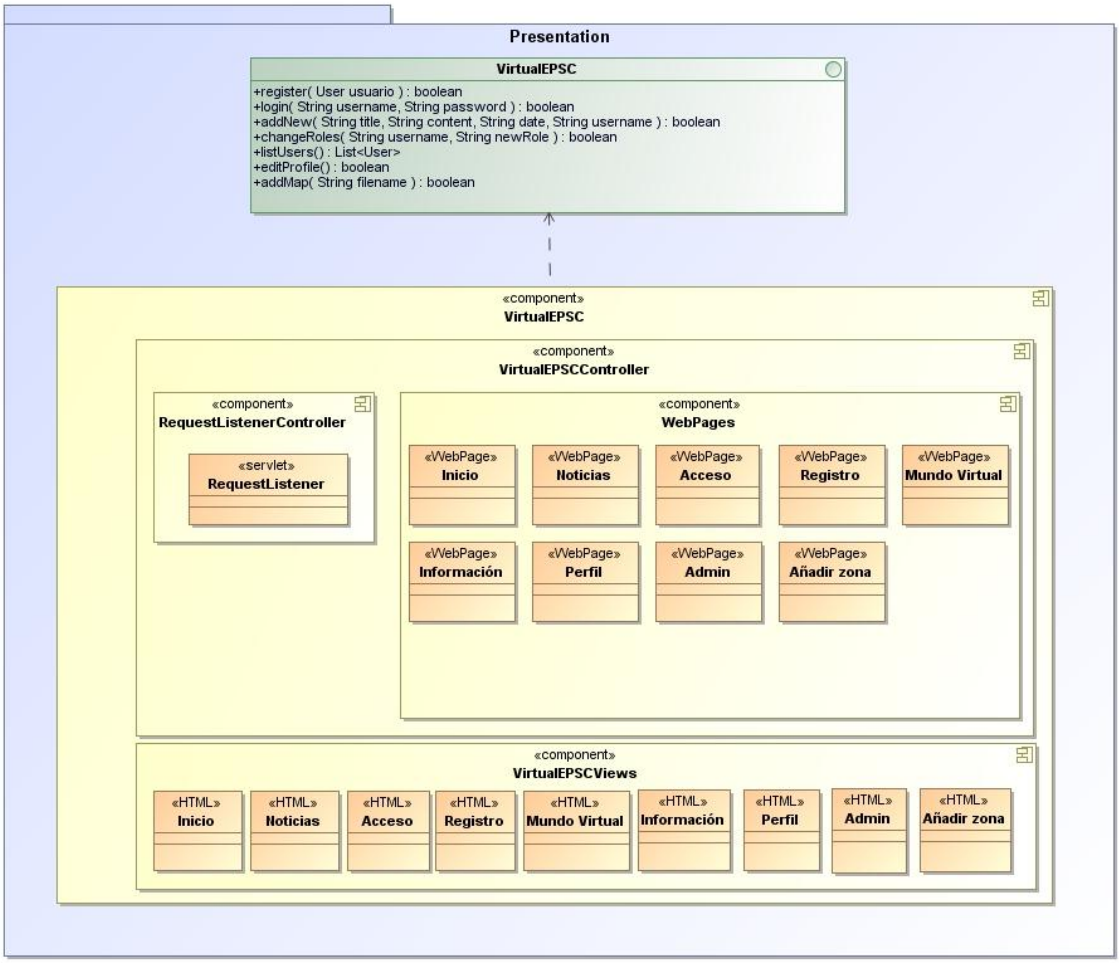

Fig. 34 Diagrama de componentes del portal web de VirtualEPSC

Como se puede observar en la figura 34, el sitio web ofrece una serie de funcionalidades definidas previamente (véase apartado 2.1.2), como son el registro y el acceso de usuario o la edición del perfil de usuario. Para ofrecer estas funcionalidades, se necesitan una serie de páginas webs que permitan interactuar con el usuario. Los encargados de esta función son los componentes de Vista y Controlador.

- Vista: son aquellos componentes que se encargan de presentar el modelo en un formato adecuado para el usuario. Hace referencia a todos los ficheros HTML de cada una de las páginas web del portal. El componente de vista está formado por los siguientes ficheros:

    - Inicio: Página principal de la aplicación desde donde se tiene acceso al resto de páginas del sitio web.
    - Noticias: Ofrece una lista de todas las noticias ordenadas por fecha que han sido añadidas a la web.
    - Acceso: Permite la autenticación y acceso de usuarios registrados en la web.
    - Registro: Permite registrar un nuevo usuario en la web.
    - Mundo Virtual: Ofrece la aplicación del mundo virtual de la EPSC, desde donde se puede visitar todas las zonas del campus y acceder a diferentes servicios de la escuela.
    - Información: Muestra información de los creadores del sitio web y de la aplicación, además de ofrecer un formulario de contacto para atender las posibles cuestiones de los usuarios.
    - Perfil: Muestra toda la información relacionada con un determinado usuario, así como su imagen de perfil.
    - Admin: Ofrece una serie de funcionalidades que solo son disponibles a aquellos usuarios que tengan roles de administradores o desarrolladores. Estas funcionalidades pueden ser asignar o quitar roles a otros usuarios y añadir noticias a la web.
    - Añadir zona: Ofrece la posibilidad que un usuario pueda añadir una nueva zona a la aplicación del mundo virtual.

- Controlador: son aquellos componentes que se encargan de responder a eventos, usualmente acciones del usuario, e invocan peticiones a la vista. Estos componentes son de tipo WebPage. La capa de Controlador está formada por los siguientes componentes:

    - Inicio: Proporciona la información de bienvenida de usuario
    - Noticias: Lista todas las noticias añadidas a la web.
    - Acceso: Realiza la autenticación de usuario.

- Registro: Registra un nuevo usuario en la web.
- Mundo Virtual: Aloja la aplicación del mundo virtual.
- Información: Proporciona información de los creadores del sitio web y de la aplicación, además de procesar el formulario de contacto.
- Perfil: Recupera toda la información relacionada con un determinado usuario, así como su imagen de perfil.
- Admin: Añade una nueva noticia y cambia el rol de un usuario determinado.
- Añadir zona: Recibe los ficheros de configuración de una zona del mundo virtual y lo aloja en un determinado directorio para que sea accesible por la aplicación.

3.3. SISTEMA DE ALMACENAMIENTO DE DATOS

Uno de los requisitos del proyecto es proporcionar un sistema de almacenamiento de datos altamente escalable frente a grandes volúmenes de datos. Se han estudiado diversas posibilidades basadas en modelos relacionales, como pueden ser MySQL y PostgreSQL.

Finalmente, se ha escogido una opción no relacional, Apache Cassandra [7], como sistema de almacenamiento de datos ya que ofrece un sistema más seguro y estable, con un acceso rápido y eficiente a los datos. Las grandes redes sociales, como Facebook, Twitter, Digg, etc. están migrando sus bases de datos a Cassandra.

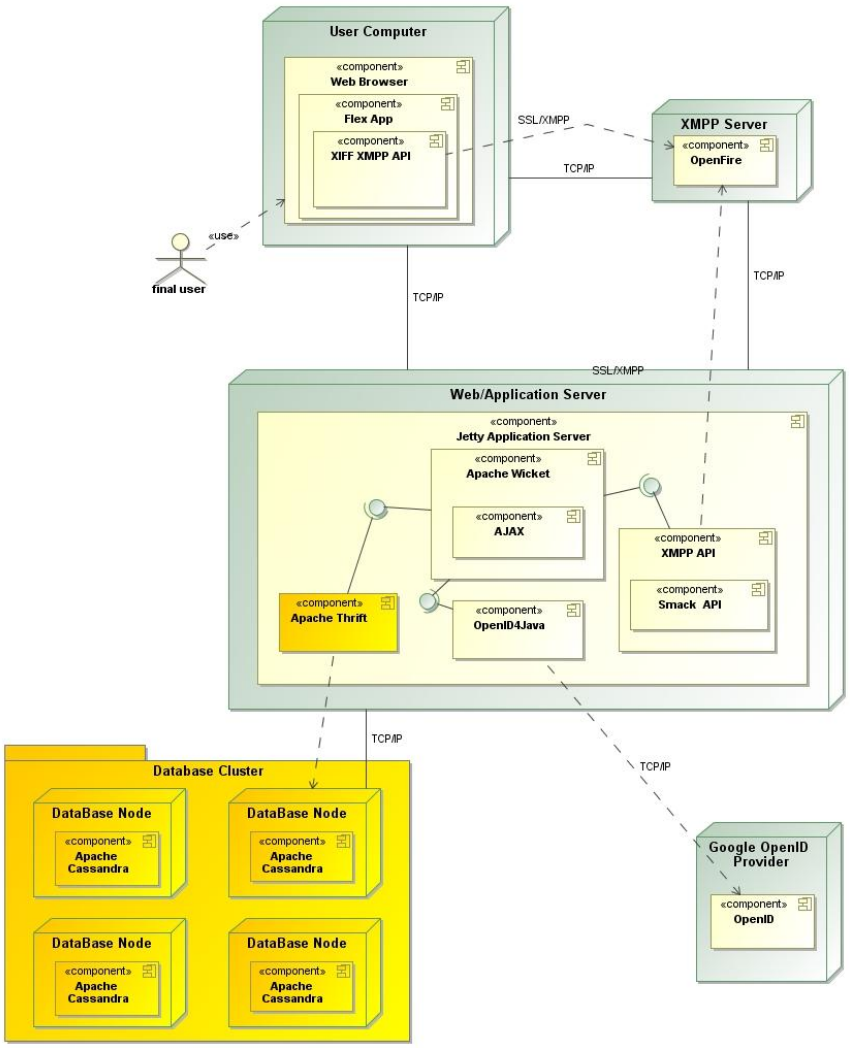

Fig. 35 Arquitectura del sistema – Sistema de almacenamiento de datos

Casssandra es un sistema de gestión de bases de datos distribuido, diseñado para manejar grandes cantidades de datos, escrito en Java y que se ha extendido gracias a su gran escalabilidad. El proyecto Cassandra es open source y basado en una arquitectura NoSQL [8], que se ha demostrado que es

mucho más eficiente que otros sistemas, como por ejemplo MySQL, en cuanto a almacenar grandes cantidades de datos [9 y 10]. Para ver más datos sobre Apache Cassandra consultar el anexo C.

3.3.1. Implementación

La implementación del sistema de almacenamiento de datos del proyecto se divide en dos apartados. Por un lado, está el diseño de la base de datos y, por otra parte, la conexión entre la base de datos y el servidor web.

3.3.1.1. Diseño de la base de datos

En cuanto al diseño de la base de datos, se ha decidido crear un KeySpace llamado VirtualEPSC que está formado por dos CollumnFamilies. Una de las ColumnFamilies se llama Usuarios, y como su nombre indica es la encargada de almacenar información referente a los usuarios. La otra ColumnFamily tiene como nombre Noticias, y se encarga de almacenar todas las noticias que se van añadiendo a la web, con toda la información complementaria.

Para ello es necesario configurar el fichero *storage-conf.xml,* que se encuentra en el directorio *conf* dentro del directorio raíz de Apache Cassandra, tal y como se define en la figura 36.

```
<Keyspace Name="VirtualEPSC">
  <ColumnFamily CompareWith="UTF8Type" Name="Usuarios" />
  <ColumnFamily CompareWith="UTF8Type" Name="Noticias" />
  <!-- Necessary for Cassandra -->
  <ReplicaPlacementStrategy>org.apache.cassandra.locator.
      RackUnaw areStrategy</ReplicaPlacementStrategy>
  <ReplicationFactor>1</ReplicationFactor>
  <EndPointSnitch>org.apache.cassandra.locator.EndPointSnitch
      </EndPointSnitch>
</Keyspace>
```

Fig. 36 Fichero storage-conf.xml de configuración de Cassandra

Principalmente, en la figura 36, se definen dos ColumnFamilies que almacenarán los dos tipos de información más importantes de la web: la de usuarios y la de las noticias.

La ColumnFamily de Usuarios está formada por parejas key-value (clave-valor). Cada clave hace referencia al nombre de usuario de cada usuario y el valor almacena todos los datos del usuario referenciado mediante la clave. Estos valores están almacenados en diferentes columnas:

- *name*: almacena el nombre real del usuario.
- *surname*: almacena los apellidos del usuario.
- *password*: almacena la contraseña del usuario.
- *score*: almacena la puntuación que tiene el usuario.
- *imageProfile*: almacena la imagen de perfil del usuario.
- *avatar*: almacena el nombre de la imagen predefinida que representará al usuario en el mundo virtual.
- *email*: almacena el correo electrónico del usuario.
- *hobbies*: almacena las aficiones del usuario. (opcional)
- *role*: almacena el rol que tiene el usuario dentro de la web.
- *list*: almacena la lista de usuarios que son amigos del usuario en cuestión.

En la figura 37, se puede observar un ejemplo de la organización de los datos en la que se organiza la ColumnFamily de Usuarios, con usuarios reales registrados en la web de VirtualEPSC.

ColumnFamily: Usuarios

Key	Value
"luismi.amoros"	**Columns** **Name** / **Value** "name" / Luismi "surname" / Amorós Martínez "password" / xxxx "score" / 2 "imageProfile" / Byte [ ] "avatar" / boy2 "email" / luismi.amoros@gmail.com "hobbies" / Fútbol "role" / DEVELOPER "list" / noemi.arbos yufera
"noemi.arbos"	**Columns** **Name** / **Value** "name" / Noemí "surname" / Arbós "password" / xxxx "score" / 3 "imageProfile" / Byte [ ]

	“avatar” girl1 “email” noemi.arbos@gmail.com “hobbies” Leer “role” DEVELOPER “list” vitual.epsc, luismi.amoros, sarcus
……..	……..

Fig. 37 Ejemplo de la ColumnFamily de Usuarios

La ColumnFamily de Noticias está formada por parejas key-value (clave-valor). Cada clave hace referencia al título de cada noticia añadida a la web y el valor almacena todos los datos de la noticia referenciada mediante la clave. Estos valores están almacenados en diferentes columnas:

- *content*: almacena el contenido de la noticia.
- *date*: almacena la fecha en la cual se agrega la noticia a la web.
- *image*: almacena una imagen adjunta a la noticia (opcional).
- *username*: almacena el nombre de usuario del usuario que añade la noticia a la web.

En la figura 38, se puede observar un ejemplo de la organización de los datos en la que se organiza la ColumnFamily de Noticias, con noticia reales añadidas a la web de VirtualEPSC.

ColumnFamily: Noticias	
Key	**Value**
“Noticia con imagen”	**Columns**

Name	**Value**
“content”	Estamos probando a añadir una noticia con una imagen adjunta. Esperamos que funcione, un saludo!!
“date”	11/05/2010 19:37:42
“image”	Byte [ ]
“username”	luismi.amoros

<table>
<tr><td></td><td></td></tr>
<tr><td>“Noticia simple”</td><td>
<table>
<tr><th colspan="2">Columns</th></tr>
<tr><th>Name</th><th>Value</th></tr>
<tr><td>“content”</td><td>Estamos probando a añadir una noticia simple. Esperamos que funcione, un saludo!!</td></tr>
<tr><td>“date”</td><td>10/05/2010 12:37:42</td></tr>
<tr><td>“username”</td><td>noemi.arbos</td></tr>
</table>
</td></tr>
<tr><td>……..</td><td>…….</td></tr>
</table>

Fig. 38 Ejemplo de la ColumnFamily de Noticias

3.3.1.2. Comunicación con el servidor web

En todas las operaciones de almacenamiento de datos, ya sea el registro de usuarios, las modificaciones de perfil o cuando se añade una nueva noticia, es necesario que el servidor web interactúe con el sistema de almacenamiento de datos. Para ello, se ha decidido utilizar Apache Thrift [11].

Apache Thrift es un conjunto de herramientas y librerías software creadas por Facebook para acelerar el desarrollo y la implementación de servicios separados eficientes y escalables. Actualmente, es un proyecto de Apache que se encuentra en continua evolución.

Su principal objetivo es permitir comunicaciones eficientes y fiables entre diferentes lenguajes de programación mediante la abstracción de algunas porciones de cada uno de ellos en una librería común. Específicamente, Apache Thrift permite a los desarrolladores definir tipos de datos e interfaces de servicio en un archivo único, utilizando un lenguaje neutral y después generar todo el código necesario para construir clientes RPC y servidores.

Actualmente soporta los siguientes lenguajes C++, Java, Python, PHP, Ruby, Erlang, Perl, Haskell, C#, Cocoa, Smalltalk y OCaml.

Por tanto, gracias a Apache Thrift y su adaptación específica para clientes de Cassandra para Java, es posible establecer una comunicación rápida y eficiente con el sistema de almacenamiento de datos.

3.4. OPENID

Hoy en día, los sitios webs requieren nombre de usuario y contraseñas para el registro y el inicio de sesión, lo que significa que millones de personas deben recordarlos para cada una de las webs en las que están registrados. En cambio, utilizando la autenticación OpenID [12], en todos los sitios webs se podría utilizar un identificador único y olvidar los múltiples registros en diferentes páginas.

Por esta razón, para facilitar el registro y el acceso a los usuarios, se ha decidido utilizar el sistema OpenID a través del proveedor externo Google. De esta manera, un usuario con una cuenta en Google, puede registrarse en la web utilizando esa misma cuenta de usuario.

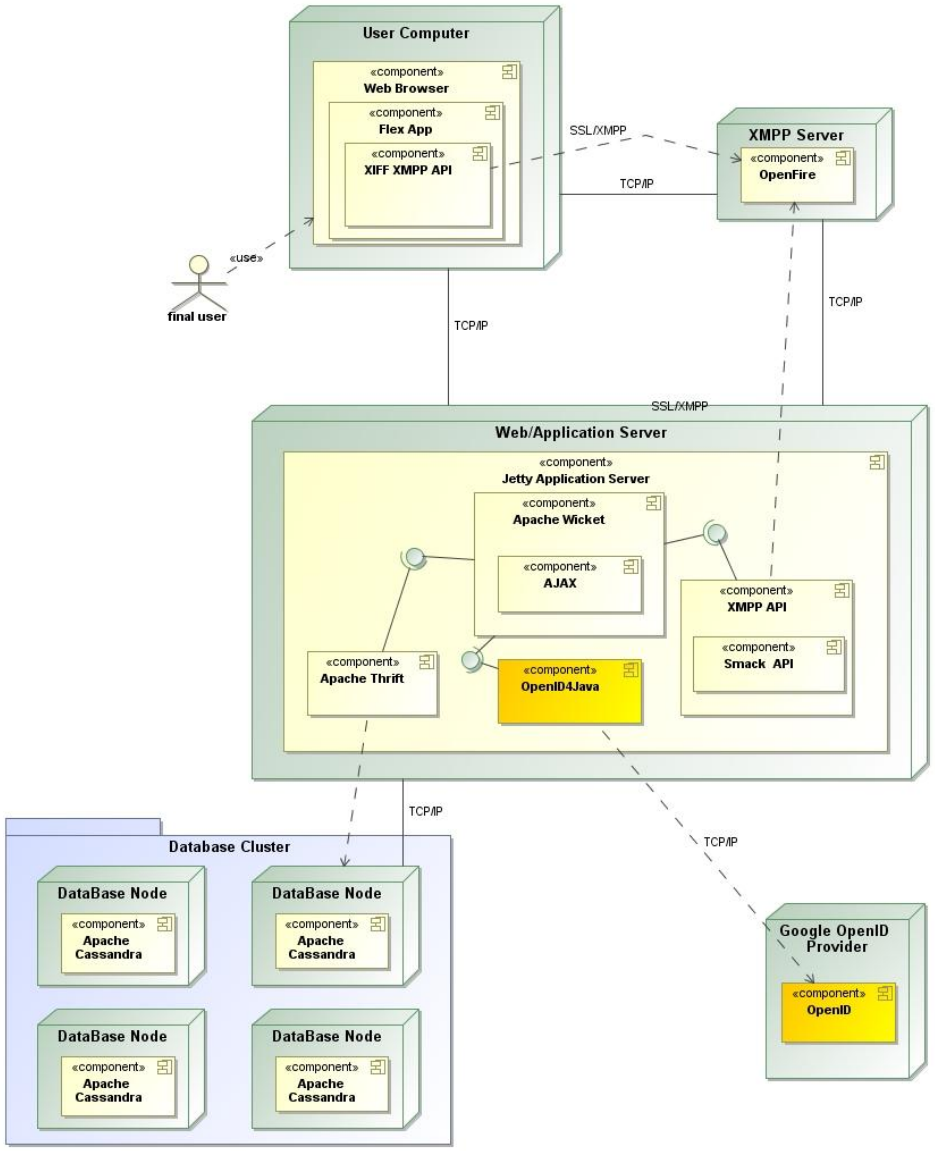

Fig. 39 Arquitectura del sistema - OpenID

OpenID es un estándar de identificación digital descentralizado, un concepto parecido a un conjunto de “servidores DNS sobre las personas”, que permite obviar la tarea de rellenar formularios de registro en las webs.

OpenID está ganando fuerza debido al anuncio de algunos grandes sitios web de la adopción de este sistema, como por ejemplo, Facebook, Google, Yahoo, MySpace, PayPal, Orange, etc. Un ejemplo de este crecimiento es que desde diciembre de 2009, hay más de mil millones de OpenIDs en Internet y aproximadamente, 9 millones de sitios web han integrado ya la autenticación OpenID. Para obtener más información sobre OpenID, consultar el anexo D.

3.4.1. Implementación

En este proyecto, para poder utilizar el servicio OpenID se ha decidido utilizar la API de OpenID4Java [13] y permitir el registro con una cuenta de Google.

OpenID4Java es una librería que permite incorporar OpenID en una aplicación Java. Utilizando esta API se puede convertir una página web en la parte confidente de una autentican OpenID.

Para utilizar Google como servidor OpenID, se ha de utilizar el identificador de Google [14], que en este caso es el mismo para todos los usuarios, *https://www.google.com/accounts/o8/id.* Como el identificador es único para todas las cuentas, el usuario ha de estar registrado en su cuenta de Google para poder completar la autenticación OpenID. De esta manera, el proveedor de identidad de Google sabe que usuario está intentando registrarse en la página de la parte confidente. Utilizando este método, Google se asegura que la comunicación se realiza con *chekid_setup*, y que por lo tanto el usuario final se tendrá que autenticar y tendrá la opción de aceptar o no la confianza con la parte confidente.

En la aplicación, se permite que el usuario se registre y se loguee en la web utilizando OpenID. El funcionamiento se divide, por lo tanto, en dos situaciones diferentes: el registro, donde se obtienen todos los datos de usuario de la cuenta de Google y se guardan en la base de datos; y el acceso, en este caso únicamente se necesita saber cuál es la cuenta de Google del usuario y comprobar que ya está registrado en la web, es decir, que sus datos están en la base de datos.

3.4.1.1. Registro

Para poder utilizar los servicios de la web el usuario debe realizar el proceso de registro. Este proceso mediante OpenID se divide en 5 pasos, tal y como se puede observar en la figura 40:

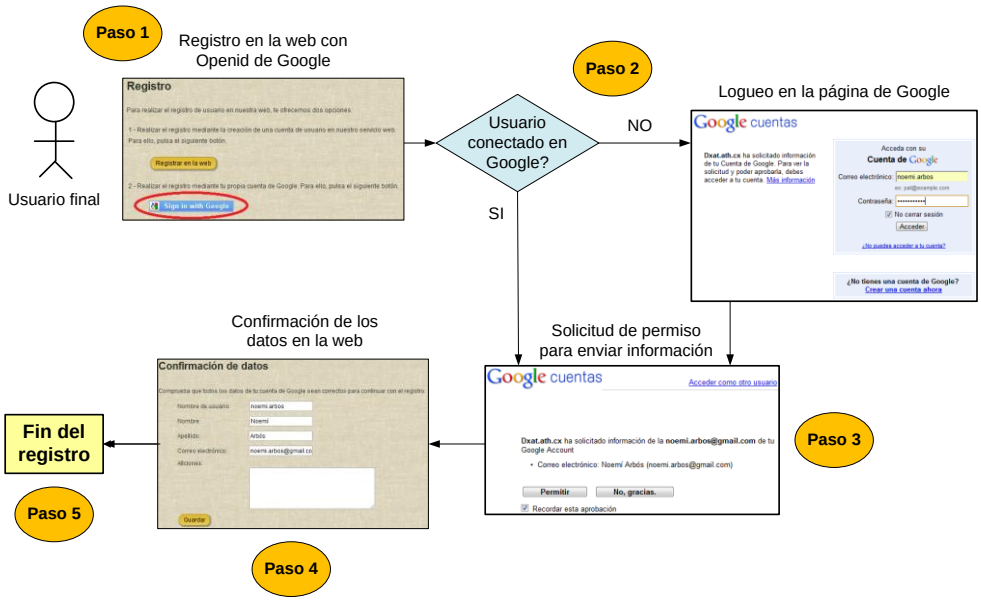

Fig. 40 Proceso de registro con OpenID

1. El usuario accede a la página de registro de la web y elige la opción de registrar mediante una cuenta de Google ya creada.

2. Se redirige al usuario a la página principal de Google y el proveedor comprueba si el usuario esta o no logueado. Si no lo está, se solicita que se loguee en la cuenta utilizando su nombre de usuario y contraseña. En esta página de logueo de Google, se informa al usuario que hay un sitio web que está solicitando información sobre su cuenta, y en el caso de que no quiera enviar esta información, simplemente puede decidir que no se acceda a su cuenta de Google.

3. Si el usuario se loguea correctamente o ya estaba logueado en su cuenta en el paso 2, al ser la primera vez que se accede a la web utilizando el identificador de Google, este proveedor redirige al usuario final a una página donde se solicita el consentimiento para enviar la información de su cuenta a esta web.

4. Si el usuario responde positivamente al consentimiento, el proveedor envía la información de la cuenta de Google a la web. En ese momento, se redirige al usuario final a un formulario, parcialmente rellenado con sus datos, que puede terminar de rellenar o simplemente aceptar el fin del registro en la web.

5. Cuando el usuario acepta el registro, sus datos son guardados en la base de datos de la aplicación.

3.4.1.2. Acceso

En el instante en que un usuario desea acceder a la web debe realizar un proceso de logueo para identificarse como usuario registrado. Este proceso mediante OpenID se divide en 5 pasos, tal y como se puede observar en la figura 41:

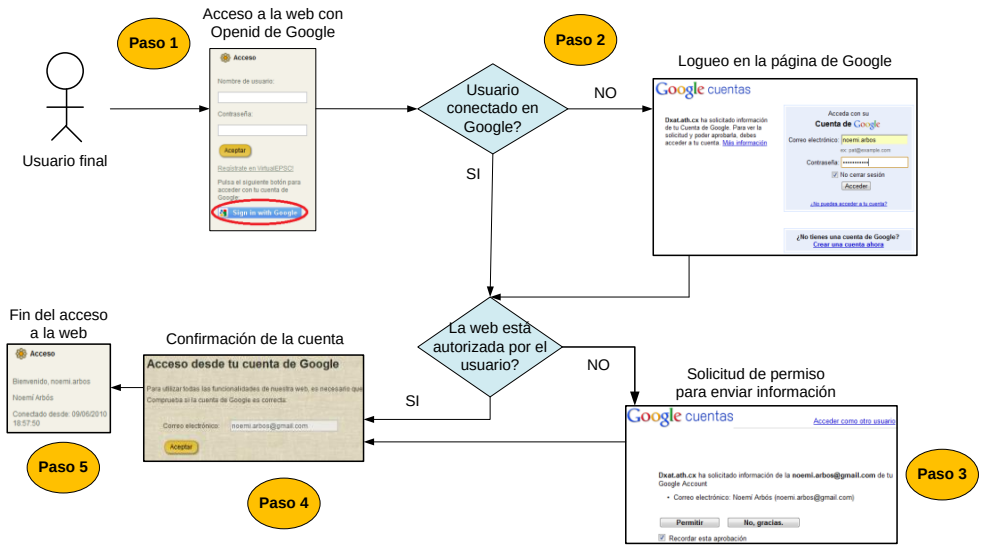

Fig. 41 Proceso de acceso con OpenID

1. El usuario quiere loguearse en la web utilizando su cuenta de Google.

2. Se redirige al usuario a la página principal de Google y el proveedor comprueba si el usuario está logueado o no. Si no lo está, se solicita que se loguee en la cuenta utilizando su nombre de usuario y contraseña. En esta página de logueo de Google se informa al usuario que hay un sitio web que está solicitando información sobre su cuenta, y en el caso de que no quiera enviar esta información, simplemente puede decidir que no se acceda a su cuenta de Google.

3. Si el usuario se loguea correctamente o ya estaba logueado en su cuenta en el paso 2, se comprueba si la web está autorizada para acceder a los datos del usuario. Si no lo está, este proveedor Google redirige al usuario final a una página donde se solicita el consentimiento para enviar la información de su cuenta a la web.

4. Si el usuario responde positivamente al consentimiento o la web ya estaba autorizada previamente, el proveedor envía la información de la cuenta de Google a la web. En ese momento, se redirige al usuario final a una página para confirmar que realmente quiere acceder a la web con esta cuenta de Google.

5. Cuando el usuario acepta el acceso a la web con la cuenta de Google, se loguea al usuario y aparecen sus datos en la parte de Acceso situada en la parte derecha de la web.

3.5. MENSAJERÍA INSTANTÁNEA

El mundo virtual ha de ser interactivo, por lo tanto, todos los usuarios han de ver lo mismo en el mismo instante de tiempo. Además, el juego cuenta con un servicio de chat público y privado.

Para alcanzar estos objetivos, la aplicación cuenta con un servicio de mensajería instantánea basado en el protocolo XMPP [15]. Extensible Messaging and Presence Protocol (*Protocolo extensible de mensajería y comunicación de presencia*), más conocido como XMPP (anteriormente llamado Jabber), es un protocolo abierto y extensible basado en XML, ideado para mensajería instantánea. Para obtener más información sobre XMPP, consultar el anexo E.

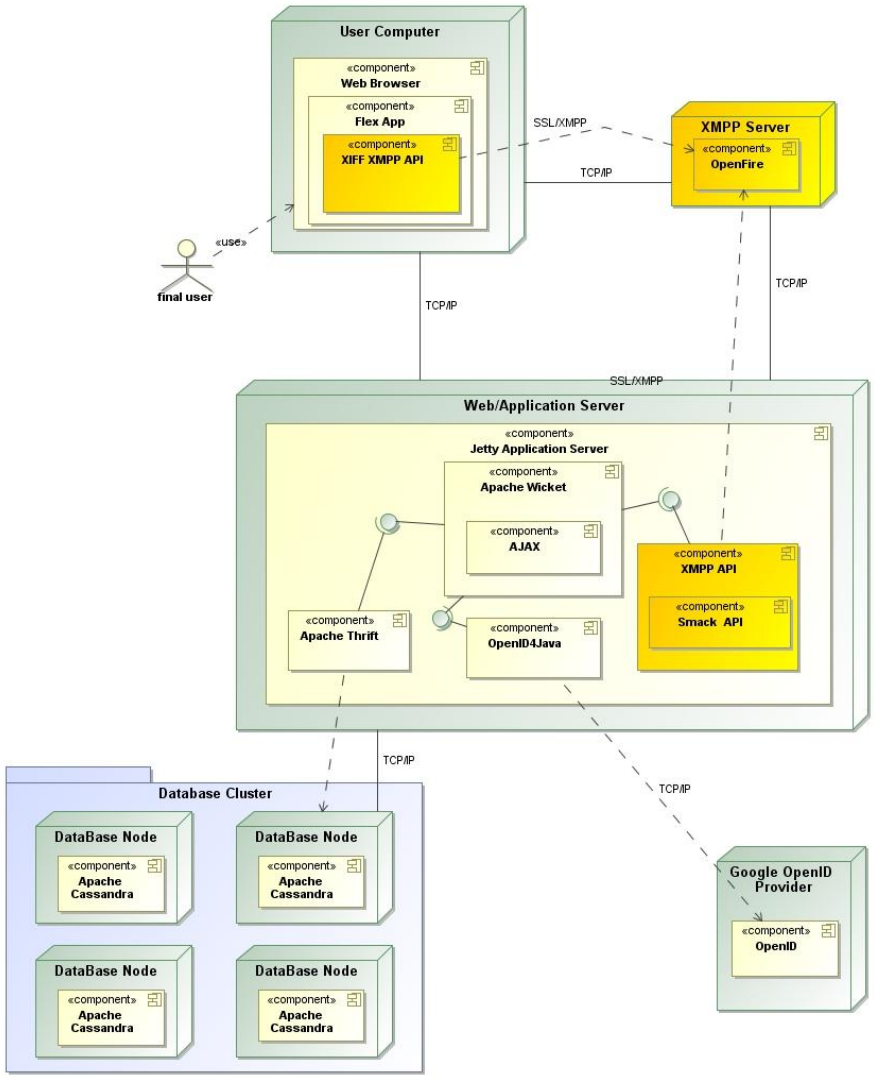

Fig. 42 Arquitectura del sistema – Mensajería instantánea

Para poder utilizar este servicio, se necesita un servidor de mensajería instantánea que utilice este protocolo, en este caso, se ha escogido Openfire [16]. El servidor Openfire es un sistema de mensajería instantánea con licencia

Open Source GPL. Está desarrollado en Java y utiliza el protocolo XMPP. Para obtener más información sobre Openfire, consultar el anexo F.

3.5.1. Implementación

La implementación de la mensajería instantánea se divide en tres apartados: la configuración del servidor Openfire, que permitirá la comunicación entre usuarios; la creación de las cuentas de usuario XMPP; y, por último, el diseño del protocolo de comunicación diseñado para señalizar los diferentes eventos de la aplicación.

3.5.1.1. Servidor Openfire

Se ha escogido utilizar el servidor Openfire como servidor de XMPP. La instalación se ha realizado sobre Linux, pero la configuración se realiza a través de la consola de administración web de Openfire, que se instala por defecto en *http://ip_servidor:9090*, si se utiliza HTTP, o en *http://ip_servidor:9091* si se quiere usar HTTPS.

Para el almacenamiento de datos del servidor Openfire, se ha escogido usar una base de datos en MySQL y alojada en *localhost*. Para poder utilizarla, se ha de instalar MySQL en el servidor y configurar el nombre de usuario y contraseña para acceder.

Cuando se configura por primera vez el servidor, se escogen las propiedades principales:

- Nombre del Dominio del servidor: *dxat*
- Puerto de la consola de administración: 9090 (por defecto)
- Puerto de la Consola de Administración Segura: *9091 (por defecto)*
- Configuración de la fuente de datos:
    - Conexión estándar, usar una base de datos externa
    - Tipo de base de datos: *MySQL*
    - Clase del Driver JDBC: *com.mysql.jdbc.Driver*
    - URL de la Base de datos: *jdbc:mysql://localhost:3306/openfire*
    - Nombre de usuario: *root*
    - Contraseña: *xxxx*
    - Número mínimo de conexiones: *5*
    - Número máximo de conexiones: *25*
    - Tiempo de la vida de la conexión: *1 día*
- Almacenamiento de usuarios y grupos en la base de datos de Openfire.

Otras características del servidor Openfire de la aplicación son:

- Clustering deshabilitado.
- Eliminación de mensajes fuera de línea.
- Auditoría de mensajes deshabilitada, no se guardarán paquetes.

3.5.1.2. Salas de conferencias

Para que los jugadores puedan enviarse mensajes entre ellos a través del servidor Openfire, en éste se han creado diferentes salas de conferencias.

Para la aplicación, existen dos tipos de salas de conferencias: las salas de zona, que permiten enviar mensajes solamente a aquellos que se encuentran en la misma zona; y la sala principal, que permite enviar mensajes a todos los usuarios conectados.

- La sala principal: su nombre es world0, a esta sala entran todos los usuarios al conectarse y permanecen en ella hasta que abandonan el juego.

  Como todos los usuarios están siempre en esta sala, si se envía un mensaje dirigido a ella, todos los usuarios conectados reciben la información. Esto sirve para enviar los mensajes de novedades de los amigos o para saber cuándo se conecta o desconecta un usuario.

- Las salas de zona: existe una sala de zona para cada zona virtual del juego, para diferencias el nombre de las salas coincide con el nombre de las zonas. Por ejemplo, existe la sala de conferencias con nombre c0f0 o la c1f1ic0f0.

  Como estas salas tienen relación con la zona de juego en la que el usuario está situado, solamente se conectan a ellas cuando acceden a esa zona, y se desconectan cuando se mueven hacia otra zona del mundo. Esta característica permite que si se envía un mensaje a una de estas salas, solo recibirán la información aquellos usuarios que se encuentren en la misma zona del juego. Se utilizan para enviar mensajes sobre el movimiento del avatar o los mensajes del chat público de zona.

Para poder crear las salas en el servidor Openfire, hace falta tener creado un Group Chat Service (Servicio de grupo de chats). En el servidor de la aplicación, existe un servicio llamado “conference” donde se encuentran todas las salas creadas. La configuración de este Group Chat Service es la siguiente:

- No mostrar histórico.
- Nunca desconectar a los usuarios.

Además, Openfire permite cambiar la configuración de cada una de las salas por separado, pero en este caso, todas tienen las siguientes características:

- Todas pertenecen al Group Chat Service “conference”
- Son salas públicas para todo tipo de usuarios.
- No son volátiles.
- El número de miembros conectados es ilimitado.
- Permiten que todos los usuarios se registren en la sala.
- No se guardan las conversaciones de la sala.

3.5.1.3. Cuentas de usuario

Para que los usuarios puedan enviar y recibir mensajes de los otros usuarios utilizando XMPP, cada uno de ellos ha de tener una cuenta creada en el servidor Openfire.

La creación de esta cuenta es totalmente transparente para el usuario, ya que se crea desde el servidor web en el momento del registro en la web, ya sea un registro normal o utilizando OpenID. El usuario creado en Openfire tiene el mismo nombre de usuario y contraseña que el creado en el registro en la página web.

Para conseguirlo se utiliza la API Smack para Java [17]. Esta API es de código abierto y proporciona un cliente de mensajería instantánea XMPP. Es una librería Java que se puede incluir en las aplicaciones para crear clientes XMPP completos o simples integraciones, como en este caso, únicamente crear las cuentas de usuario.

Por otra parte, cuando se accede a la pestaña del mundo virtual en la web, se envía el nombre de usuario y la contraseña del usuario conectado. De esta manera, desde la aplicación Flash se podrá utilizar otro cliente XMPP para poder enviar y recibir mensajes.

En Flex se ha utilizado la API XIFF [18]. Esta API, también es de código libre y proporciona un cliente de mensajería instantánea XMPP, en este caso para aplicaciones Flash.

Por lo tanto, utilizando XIFF, desde la aplicación los usuarios se conectarán al servidor Openfire utilizando su cuenta creada anteriormente, y podrán enviar y recibir mensajes a las salas de conferencias del servidor que estén conectados o únicamente a otro usuario.

3.5.1.4. Protocolo de comunicación

Para que el juego sea interactivo, es decir, que los usuarios puedan ver dónde se encuentran los avatares del resto de usuarios en cada momento y poder hablar a través de chats públicos y privados, se ha diseñado un protocolo de mensajes entre usuarios.

Cada uno de los mensajes avisa al resto de jugadores de un evento generado por el usuario que está enviando el mensaje, ya sea únicamente a los que se encuentren en la misma zona del mundo virtual o a todos los que estén conectados en ese momento.

Para ello, los usuarios se encuentran conectados a dos salas diferentes del servidor. Una de ellas es la sala general, llamada *world0*, donde los usuarios están conectados desde que inician la aplicación hasta que la abandonan. Por lo tanto, todos los mensajes enviados a esta serán recibidos por todos los usuarios que se encuentren utilizando la aplicación. Además, cada usuario está

conectado a una sala dependiendo de la zona del juego donde se encuentre. De esta manera, es muy sencillo enviar un mensaje únicamente a los usuarios que se encuentran en la misma zona. Estas zonas tienen el nombre de las zonas del juego, por ejemplo *worldc0f0* o *worldc1f1ic0f0*.

También existe la posibilidad de enviar mensajes privados, por ejemplo, cuando se establece una conversación mediante un chat privado con otro usuario. Para conseguirlo, simplemente hay que enviar el mensaje a ese usuario, en lugar de enviarlo a las salas.

Todos los mensajes que se envían están basados en XML para que sea más fácil su creación a partir del evento y permitir distinguir un tipo de mensajes de los otros.

A continuación, se muestran los diferentes tipos de mensajes:

Tabla 3 Tipos de mensajes

TIPO	MENSAJE	DESCRIPCIÓN
Connect	<event> <general></general> </event>	Indica a la sala general que un usuario se ha conectado.
Connect Reply	<event> <connectReply></connectReply> </event>	Respuesta a un mensaje Connect. Se envía a la sala general.
Presence	<event> <presence> <image>***avatar***</image> </presence> </event>	Indica a los usuarios de una zona que un usuario ha entrado. Se envía a la sala de la zona.
Presence Reply	<event> <presence> <image>***avatar***</image> <pointX>***posiciónX***</pointX> <pointY>***posiciónY***</pointY> </presence> </event>	Respuesta a un mensaje Presence. Se envía a la sala de la zona.
Leave	<event> <leave></leave> </event>	Indica que un usuario abandona una zona. Se envía a la sala de la zona.
Disconnect	<event> <disconnect></disconnect> </event>	Indica que un usuario se desconecta del mundo virtual. Se envía a la sala general.

Move	<event> <move> <pointX>***posiciónX***</pointX> <pointY>***posiciónY***</pointY> </move> </event>	Indica que un usuario se está moviendo. Se envía a la sala de la zona.
Group Message	<event> <groupMessage> <msg>***texto***</msg> </groupMessage> </event>	Es un mensaje de chat público. Se envía a la sala de la zona.
Info Friend	<event> <infoFriend>***texto del evento***</infoFriend> </event>	Indica una novedad de un usuario. Se envía a la sala general.
Create Private Chat	<event> <createPrivateChat></createPrivateChat> </event>	Indica que un usuario quiere establecer un chat privado con otro. Se envía al usuario.
Text Private Chat	<event> <textPrivateChat>***texto***</textPrivateChat> </event>	Es un mensaje de chat privado. Se envía al usuario.

Hay dos tipos de mensajes más, pero únicamente, se utilizan en el momento en que un usuario decide darse un paseo en barca por el lago del mundo virtual.

Tabla 4 Tipos de mensajes para el paseo en el lago

TIPO	MENSAJE	DESCRIPCIÓN
Lake	<event> <paseoLago> ***avatar*** </paseoLago> <pointX>***posiciónX***</pointX> <pointY>***posiciónY***</pointY> </event>	Indica que un usuario se mete en el lago. Se envía a la sala de la zona.
Lake Stop	<event> <paseoAcabaLago>***avatar*** </paseoAcabaLago> <pointX>***posiciónX***</pointX> <pointY>***posiciónY***</pointY> </event>	Indica que un usuario sale del lago. Se envía a la sala de zona.

En una situación como la de la figura, donde un usuario (Toni) entra en la sala, mientras otro (Luismi) está dentro del lago, se produce un caso especial en la respuesta al mensaje Presence. Como se puede ver en la figura 43, el usuario

que está en el lago, Luismi, responde con un mensaje de tipo Lake, en lugar de un mensaje PresenceReplay. De esta manera, el usuario Toni se da cuenta de que está dentro del lago y, por lo tanto, hay que mostrarlo con la barca.

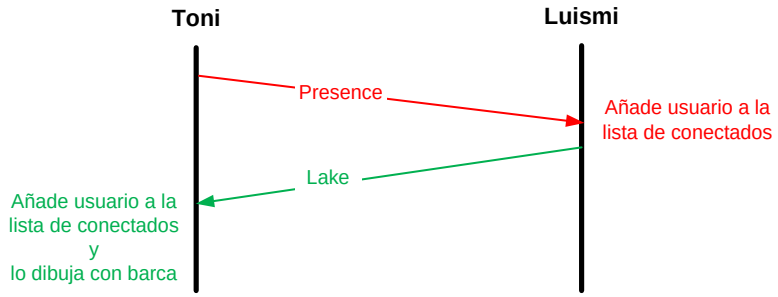

Fig. 43 Uso del mensaje Lake como respuesta a un mensaje Presence

Para saber qué jugador ha enviado el mensaje, utilizamos la propiedad *from*, es decir, quién es el emisor del mensaje. De esta manera, se consigue que no se pueda suplantar a un jugador manipulando el contenido del texto del mensaje. Por ejemplo, si se mandara dentro del mensaje una variable *user* que indicara quién es el emisor, desde un cliente XMPP se podría crear el mensaje cambiando esa variable sin que sea realmente ese usuario el que envía los mensajes.

Ejemplo de suplantación, donde el resto de usuarios cree que se ha conectado Pepe:

Alicia envía: <event><general><user>Pepe</user></general></event>

Todos los usuarios reciben los mensajes que son enviados a la sala general y a la sala donde están conectados. Pero no todos los usuarios han de actuar de la misma manera ante un mismo mensaje. Los diferentes casos son los siguientes:

- *Se conecta un nuevo usuario al mundo virtual.*

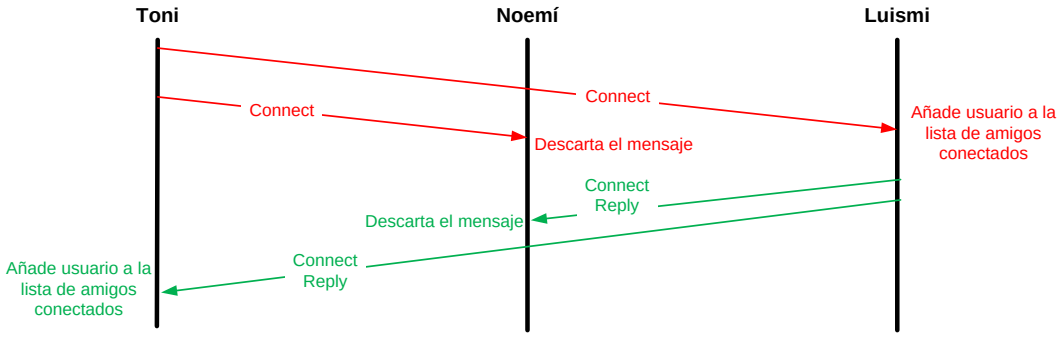

Fig. 44 Protocolo de mensajes cuando se conecta un nuevo usuario

En la figura 44, se puede ver cuál es el proceso de envío de mensajes cuando un usuario se conecta. Los pasos que se siguen son los siguientes:

1. Noemí y Luismi ya están conectados. El nuevo usuario, Toni, al entrar envía un mensaje Connect a la sala general.

2. Los usuarios Noemí y Luismi reciben el mensaje Connect de Toni, y comprueban si tienen al usuario Toni en su lista de amigos. Como Noemí no lo tiene agregado como amigo, descarta el mensaje. En cambio Luismi, que sí lo tiene en su lista, añade a Toni en la lista de amigos conectados.

3. El usuario Luismi envía un mensaje ConnectReply a la sala general para contestar al mensaje Connect del nuevo usuario. Tanto Toni como Noemí reciben ese mensaje. Toni comprueba si tiene a Luismi en su lista de amigos, si lo está, lo agrega a su lista de amigos conectados. Noemí puede descartar el mensaje porque Luismi no está en su lista de amigos o porque ya está agregado en su lista de amigos conectados.

- *Entra un nuevo usuario en una zona.*

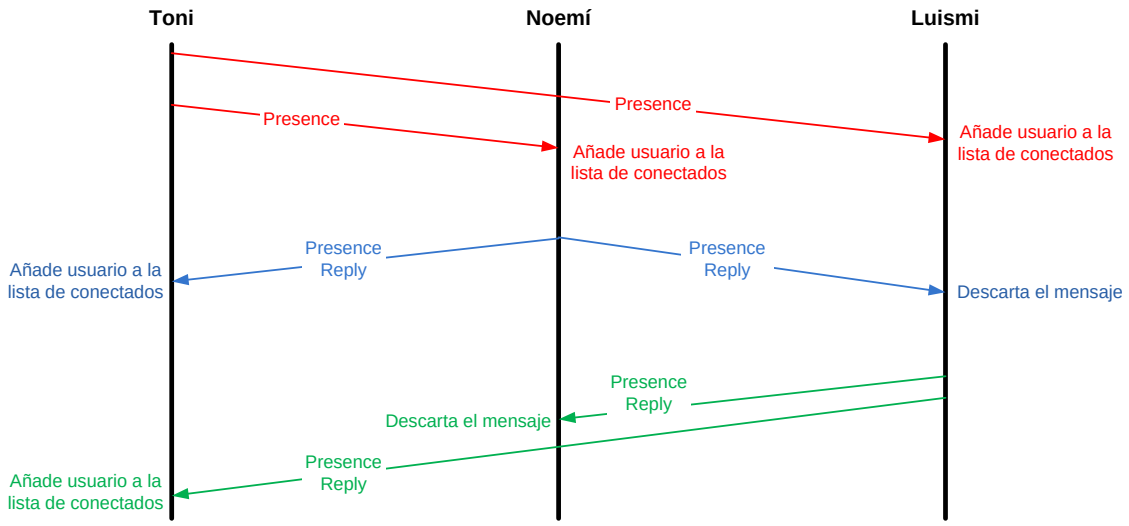

Fig. 45 Protocolo de mensajes cuando un usuario entra en una zona

En la figura 45, se puede ver cuál es el proceso de envío de mensajes cuando un usuario entra en una nueva zona. Los pasos que se siguen son los siguientes:

1. Noemí y Luismi ya están dentro de la zona. El nuevo usuario, Toni, al entrar envía un mensaje Presence a la sala.

2. Los usuarios Noemí y Luismi reciben el mensaje Presence de Toni, comprueban que no tienen al usuario en la lista de conectados a la sala y lo añaden.

3. El usuario Noemí envía un mensaje PresenceReply para contestar al mensaje Presence del nuevo usuario. Tanto Toni como Luismi reciben ese mensaje. Toni comprueba que no tiene a Noemí en la lista de usuarios conectados y la añade. Luismi comprueba si tiene a Noemí en la lista, como ella ya se encontraba en la sala descarta su mensaje.

4. El usuario Luismi realiza el mismo proceso que Noemí en el punto 3. En este caso, Toni lo añade porque no está en su lista, y Noemí descarta el mensaje porque ese usuario ya estaba en la misma zona.

- *Un usuario envía un mensaje InfoFriend.*

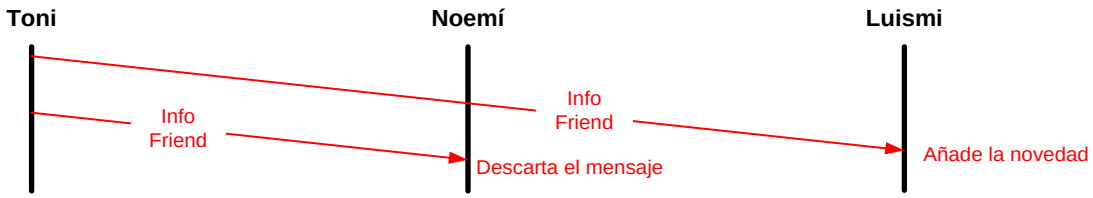

Fig. 46 Protocolo de mensajes cuando un usuario envía una novedad

En la figura 46, se puede ver cuál es el proceso de envío de los mensajes de novedades de los usuarios, por ejemplo cuando un usuario juego al ping-pong o da un pasea en barca por el lago. Los pasos que se siguen son los siguientes:

1. El usuario Toni envía un mensaje Info Friend a la sala general porque ha realizado algún evento en el mundo virtual que envía una novedad.

2. Los usuarios Noemí y Luismi reciben el mensaje Info Friend de Toni, y comprueban si tienen al usuario Toni en su lista de amigos. Como Noemí no lo tiene agregado como amigo, descarta el mensaje. En cambio Luismi, que si lo tiene en su lista, añade el mensaje en su lista de novedades.

3.6. APLICACIÓN DEL MUNDO VIRTUAL

Para el desarrollo de la plataforma del mundo virtual, se ha decidido utilizar Adobe Flex [19], que se basa en el lenguaje MXML y en lenguaje ActionScript [20] para crear aplicaciones basadas en Flash.

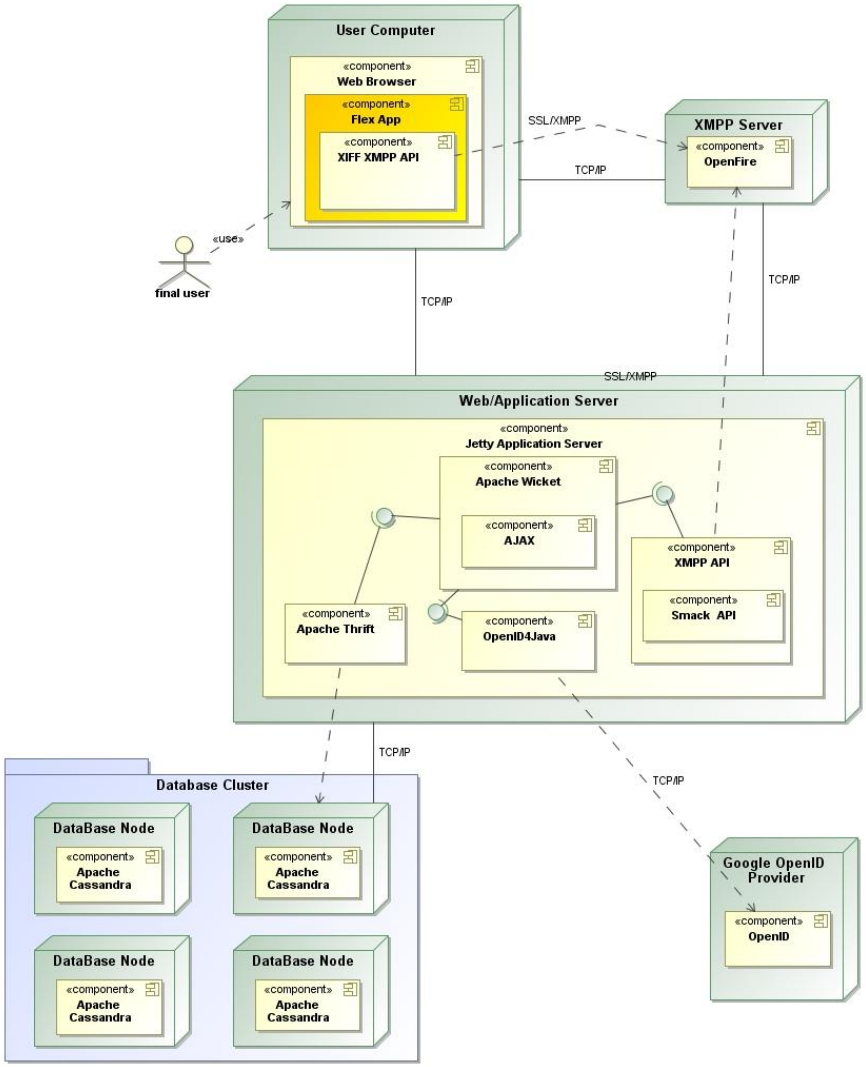

Fig. 47 Arquitectura del sistema – Aplicación del mundo virtual

El objetivo de Flex es permitir a los desarrolladores de aplicaciones web construir de manera rápida y fácilmente Aplicaciones de Internet Ricas, también llamadas RIAs. En un modelo multi-capa, las aplicaciones Flex son el nivel de presentación. Para obtener más información sobre Adobe Flex, consultar el anexo G.

Las alternativas a Flex son, entre otras, Google Web Toolkit, JavaFX, OpenLaszlo, Silverlight de Microsoft y HTML5.

3.6.1. Implementación

En la aplicación del mundo virtual, existen tres puntos principales de implementación: los estados de la aplicación, el método de comunicación con el sistema de almacenamiento de datos y los movimientos del avatar.

3.6.1.1. Estados de la aplicación

En las aplicaciones Flex se pueden definir diferentes estados, que son diferentes capas donde se define en cada una de ellas cuáles son los elementos visibles.

En la aplicación del mundo virtual existen 4 estados:

- **Join**: es el estado con el que arranca la aplicación. Simplemente contiene una etiqueta de bienvenida. Antes de pasar al siguiente estado se recogen los parámetros de nombre de usuario, contraseña y avatar, y se carga el archivo *properties.xml*, que contiene la información sobre el servidor XMPP y el servidor web.

  Dependiendo del valor del parámetro avatar se pasara al siguiente estado:

  - Si el nombre de avatar es *false,* esto indica que el usuario no tiene asignado ningún "alter ego" y, por lo tanto, antes de pasar al mundo virtual tiene que escogerlo en el estado "Elegir Personaje".
  - Si el nombre de avatar es diferente de *false*, indica cuál es el nombre del "alter ego" que el usuario tiene asignado y, por lo tanto, se pasa al estado "Game" para entrar en el mundo virtual.

- **Elegir personaje**: se pasa a este estado cuando el usuario no tiene ningún avatar asignado, es decir, la primera vez que se accede al mundo virtual. En este estado hay una imagen que muestra cómo es el avatar seleccionado; también hay dos botones que permiten ver el resto de avatares disponibles; y por último, un botón de Aceptar que asigna el avatar seleccionado al usuario y lo envía al siguiente estado: "Game".

  Para saber cuántos avatares están disponibles y cuáles son, se utiliza el fichero *personajes.xml*. En este fichero XML están definidas todas las imágenes que contiene la aplicación y que los usuarios pueden elegir como "alter ego" en el mundo virtual. El nodo principal del XML es <images></images>, y dentro de él se define cada avatar tal y como muestra la figura 48.

```
<image>
  <name>nombre del avatar</name>
</image>
```

Fig. 48 Ejemplo de definición de un avatar

- **Game**: éste es el estado donde se juega con el “alter ego” en el mundo virtual. Está formado por dos menús:

    - En el menú superior, hay un texto con el nombre de usuario, el botón para acceder al miniperfil del juego, el botón para ver el Top5, el botón para ver la lista de amigos conectados, el botón para ver las novedades, el botón para acceder al mapa del campus y el botón de salir para abandonar el juego.
    - En el menú inferior, hay una lista de todas las zonas a las que se puede acceder, el botón para ver el chat público de cada sala, y también en este menú van apareciendo los chats privados con otros usuarios.

    El otro componente de este estado es un Canvas donde se visualizan las diferentes zonas virtuales del campus y los “alter egos” del resto de usuarios conectados. Dentro del Canvas se puede mover el avatar clicando en la posición donde se quiera desplazar, aunque hay zonas donde el avatar no puede entrar. Además, en cada zona se pueden encontrar diferentes botones para acceder a otros mundos, para abrir páginas webs o abrir juegos.

    Desde este estado se puede acceder al estado de Elegir Personaje si se clica en el botón Cambiar Avatar dentro del miniperfil de usuario, o pasar al estado Abandono clicando en el botón Salir del menú superior.

- **Abandono**: este estado se utiliza para desconectar al usuario del juego y poder avisar al resto de usuarios de su abandono. Se compone únicamente de un texto informativo para que el usuario sepa que todo ha ido correctamente.

La figura 49 muestra un diagrama de estados de cómo se produce la transición de un estado a otro de la aplicación, estos cambios se pueden producir dependiendo del valor de alguna variable o clicando en los botones de la aplicación.

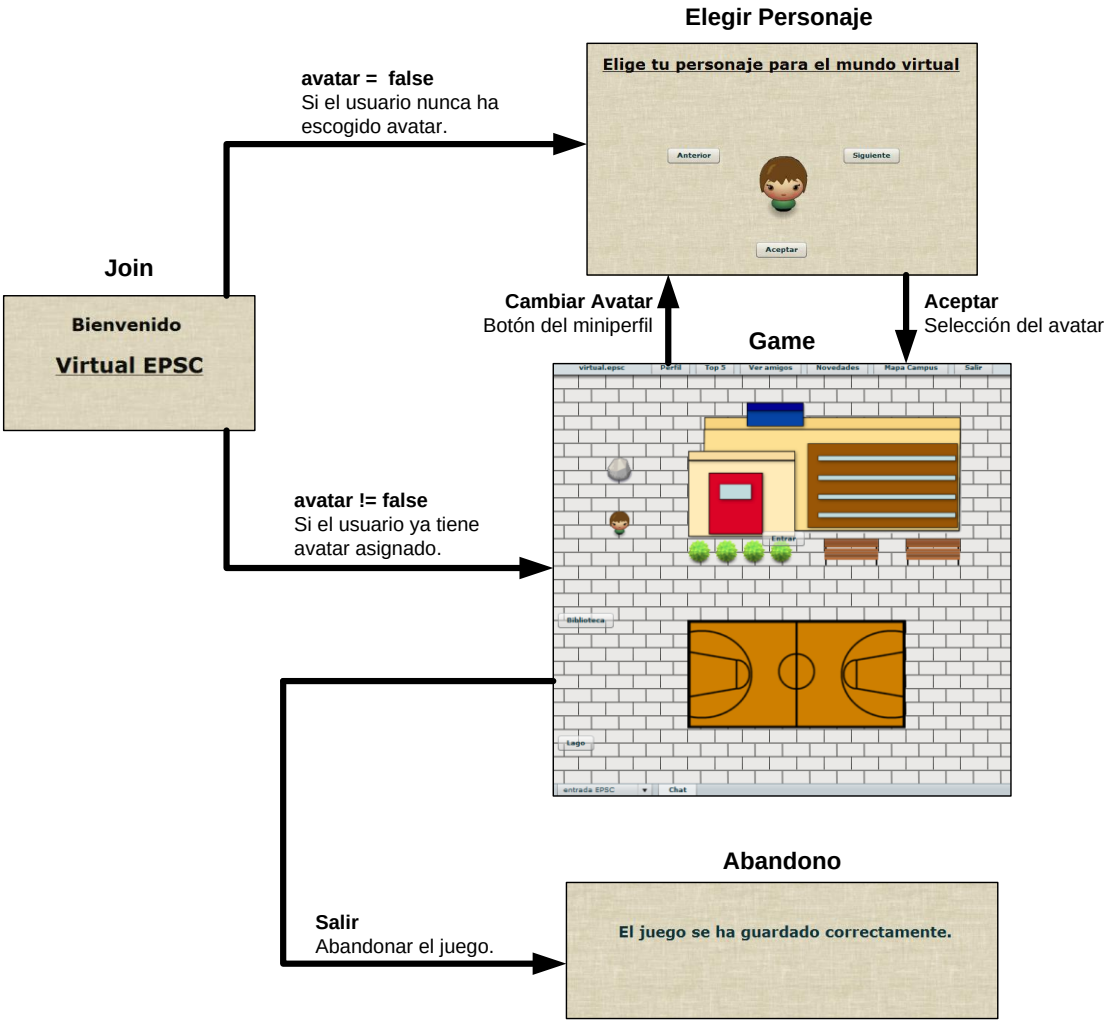

Fig. 49 Esquema de estados de la aplicación Flex

3.6.1.2. Comunicación con la base de datos

En la aplicación existen varias acciones que necesitan interactuar con la base de datos, ya sea para hacer una consulta o para cambiar algún valor. Para poder solucionar este problema se ha decidido utilizar un Servlet, que se aloja en la web y utiliza la API Thrift para comunicarse con la base de datos (véase apartado 3.3.1.2).

Por otro lado, desde la aplicación se utilizan objetos HTTPService. Estos componentes permiten hacer peticiones a los Servlets pasándoles diferentes parámetros y recibir una serie de datos como respuesta, de esta manera se consigue interactuar con la base de datos a través del Servlet de la web. En la siguiente figura se puede observar cuál es el proceso de comunicación entre la aplicación y la base de datos:

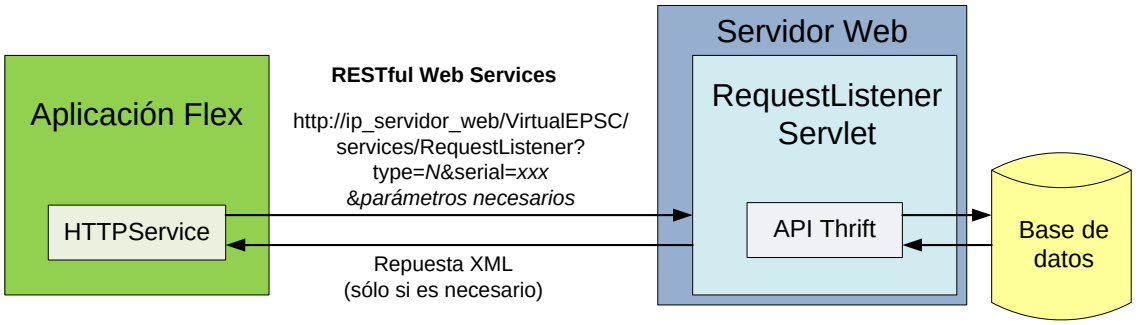

Fig. 50 Interacción entre la aplicación Flex, el Servlet y la base de datos

Desde el HTTPService de la aplicación se llama a la URL del Servlet, y se le indican una serie de parámetros necesarios. El parámetro más importante es el *type*, simplemente es un número que indica cual es la acción que se está pidiendo realizar. Otro parámetro que siempre aparece es el *serial*, un número aleatorio que indica que es una nueva petición.

Las diferentes acciones que necesitan interactuar con la base de datos son las siguientes:

- *Guardar el nombre del avatar escogido*: se utiliza cuando el usuario escoge un avatar o lo cambia, de esta manera la próxima vez que se conecte, se cogerá de la base de datos cual es su "alter ego" y evitar que tenga que volver a escoger.

  Los parámetros necesarios para poder realizar la petición son:

  - Type = 0: indica que es una acción de guardar el avatar.
  - Username: es el nombre de usuario que hay que modificar.
  - Image: es el nombre del avatar escogido.
  - Serial: número aleatorio.

  Ejemplo:

  ```
  http://ip_servidor_web/VirtualEPSC/services/RequestListener?type=0&username=noemi&image=girl3&serial=0406190123
  ```

- *Subir puntuación*: se utiliza en los juegos para subir la puntuación del usuario con los nuevos puntos conseguidos.

  Los parámetros necesarios para poder realizar la petición son:

  - Type = 2: indica que es una acción de subir puntuación.
  - Username: es el nombre de usuario que hay que modificar.
  - Score: el número de puntos conseguidos en el juego.
  - Serial: número aleatorio.

Ejemplo:

```
http://ip_servidor_web/VirtualEPSC/services/RequestListener?type=2&username=noemi&score=3&serial=0406190123
```

- *Recibir los datos de perfil de un usuario*: se utiliza al hacer una consulta de miniperfil de usuario, ya sea el propio del usuario o de otro jugador.

  Los parámetros necesarios para poder realizar la petición son:

  - Type = 3: indica que es una acción de consulta de perfil.
  - Username: es el nombre de usuario que se quiere consultar.
  - Serial: número aleatorio.

  Ejemplo:

```
http://ip_servidor_web/VirtualEPSC/services/RequestListener?type=3&username=noemi&serial=0406190123
```

  A esta petición el Servlet devuelve una respuesta en formato XML con los datos del usuario. El XML recibido tiene el siguiente formato:

```
<user>
  <perfil>
   <name>nombre completo</name>
   <score>puntuación</score>
   <image>nombre del avatar</image>
  </perfil>
</user>
```

Fig. 51 Ejemplo de respuesta a una consulta de perfil

- *Recibir la lista de los 5 mejores jugadores*: esta acción se utiliza al hacer la consulta del Top5 del juego, sirve para ver cuáles son los 5 jugadores con más puntuación.

  Los parámetros necesarios para poder realizar la petición son:

  - Type = 4: indica que es una acción de consulta del Top5.
  - Serial: número aleatorio.

  Ejemplo:

```
http://ip_servidor_web/VirtualEPSC/services/RequestListener?type=4&serial=0406190123
```

A esta petición el Servlet devuelve una respuesta en formato XML con los nombres y la puntuación de los 5 mejores usuarios. El XML recibido tiene el siguiente formato:

```
<bests>
  <name>nombre usuario del jugador 1</name>
  <score>puntuación del jugador 1</score>
  <name>nombre usuario del jugador 2</name>
  <score>puntuación del jugador 2</score>
  <name>nombre usuario del jugador 3</name>
  <score>puntuación del jugador 3</score>
  <name>nombre usuario del jugador 4</name>
  <score>puntuación del jugador 4</score>
  <name>nombre usuario del jugador 5</name>
  <score>puntuación del jugador 5</score>
</bests>
```

Fig. 52 Ejemplo de respuesta a una consulta de Top5

- *Recibir la lista de amigos*: esta acción se utiliza al hacer la consulta la lista de amigos conectados.

  Los parámetros necesarios para poder realizar la petición son:

  - Type = 5: indica que es una acción de consulta de la lista de amigos.
  - Username: es el nombre de usuario que se quiere consultar.
  - Serial: número aleatorio.

  Ejemplo:

```
http://ip_servidor_web/VirtualEPSC/services/RequestListener?type=4&username=noemi&serial=0406190123
```

  A esta petición, el Servlet devuelve una respuesta en formato XML con los nombres de los amigos. El XML recibido tiene el siguiente formato:

```
<lista>
  <amigo>nombre usuario del amigo 1</amigo>
  <amigo>nombre usuario del amigo 2</amigo>
  <amigo>nombre usuario del amigo 3</amigo>
      [...]
  <amigo>nombre usuario del amigo N</amigo>
</lista>
```

Fig. 53 Ejemplo de respuesta a una consulta de la lista de amigos

- *Añadir un usuario a la lista de amigos*: se envía esta acción ala gregar a otro jugador como amigo.

  Los parámetros necesarios para poder realizar la petición son:

  - Type = 5: indica que es una acción de consulta de la lista de amigos.
  - Username: es el nombre de usuario que se quiere consultar.
  - Friendname: es el nombre de usuario del amigo a añadir.
  - Serial: número aleatorio.

  Ejemplo:

```
http://ip_servidor_web/VirtualEPSC/services/RequestListener?type=4&username=noemi&friendname=luismi&serial=0406190123
```

3.6.1.3. Movimientos del avatar

En un mundo virtual como el de la aplicación, es de vital importancia que cada avatar o “alter ego” sea capaz de moverse libremente dentro del espacio de cada una de las zonas virtuales. Pero también es muy importante que los avatares no puedan moverse por lugares que en un mundo real no podrían pisar, como es el caso de los edificios, los arboles, etc.

Para que este movimiento sea lo más eficiente posible y permita encontrar el camino más corto entre dos puntos, se ha decidido utilizar una API llamada Tweening Platform de Greensock [21], que se basa en un algoritmo conocido como Algoritmo de búsqueda A* [22].

Este algoritmo es un algoritmo de optimización de cambios de estado. En él, se examina un sistema en su estado actual y se determina la serie de cambios de estado de menor coste necesarios para llegar a un estado final. Se suele utilizar para encontrar un camino entre nodos, pero también puede ser usado para resolver el Cubo de Rubik o puzles deslizantes. Para ver más información sobre este algoritmo consultar el anexo H.

En este proyecto, los avatares se desplazan sobre un mapa de “tiles” o cuadriculas. Por tanto, la implementación del algoritmo es mucho más sencilla que en otras superficies.

Básicamente, el objetivo del movimiento de los avatares es conseguir el camino más corto entre dos puntos A y B, pero si entre esos dos puntos existe un lago y un edificio es necesario bordearlos. El coste de atravesar un edificio o un lago, viene definido en los ficheros XML de definición de zonas del mundo virtual. Concretamente, cuando se define una zona de colisión (véase apartado 2.2.1.2), lo que se hace es asignar un coste muy alto a esa zona.

Por tanto, cuando el algoritmo se ejecuta intenta evitar dichas zonas de colisiones, ya que les producen un coste extremadamente alto en comparación a atravesar otras zonas del mapa como pueden ser hierba o baldosas. En la figura 54 podemos ver un ejemplo: como las vías están dentro de una zona de colisión y por lo tanto tienen un coste muy alto, el algoritmo decide que para atravesarlas es más fácil pasar por el puente.

Fig. 54 Ejemplo del movimiento del avatar

Para notificar el movimiento de un avatar al resto de los usuarios que se encuentran en la misma zona del mundo virtual se envía un mensaje XMPP específico (véase apartado 3.5.1.4). Una vez recibido este mensaje, se calcula el camino para dicho avatar mediante la ejecución del algoritmo. De esta manera, todos los usuarios que se encuentran en la misma zona del mundo virtual pueden calcular el mismo movimiento ya que todos tienen las mismas zonas de colisión y ejecutan el mismo algoritmo. Por tanto, a vistas de usuario, cuando un avatar se mueve, el resto observan el mismo movimiento con un retardo que viene determinado por el retardo de la mensajería instantánea que suele ser inferior a 150 ms.

3.7. APIS

Para fomentar la participación y colaboración de los usuarios (inteligencia colectiva) en la aplicación, se ponen a su disposición una serie de APIs que pueden utilizar. La primera de ellas, da la posibilidad de crear una nueva zona del mundo virtual y añadirla en la aplicación. La segunda, permite subir la puntuación de los usuarios en nuevos juegos que se pueden añadir en las nuevas zonas del mundo virtual.

3.7.1. Añadir una nueva zona al mundo virtual

Se necesita subir un archivo *.zip* en la página de añadir mapa (*AddMap*). Este archivo debe contener las siguientes partes:

- *El archivo XML que describa la zona:* el nombre de este archivo ha de seguir la forma world*nombre_del_mundo.*xml*.* Por ejemplo, si se está añadiendo la zona exterior C1F1 el nombre de la imagen será worldc1f1.xml, si se añade la zona interior C2F2 que está dentro de la exterior C1F1, el nombre será worldc1f1ic2f2.xml.

  El fichero XML ha de contener todos las tags obligatorios para que se pueda visualizar la zona. Estos tags están descritos en el apartado 2.2.1.1 de este documento.

  La figura 55 muestra un ejemplo de fichero XML que describe cómo es la zona C2F1:

```
<?xml version="1.0" encoding="UTF-8" ?>
<world>
	<name>entrada EPSC</name>
	<ground>
		<source>losa</source>
		<width>40</width>
		<height>40</height>
	</ground>
	<baldosas>
		<build>
			<tileX>5</tileX>
			<tileY>9</tileY>
			<source>basquet</source>
			<width>320</width>
			<height>160</height>
		</build>
		<build>
			<tileX>10</tileX>
			<tileY>6</tileY>
			<source>banco</source>
			<width>80</width>
			<height>40</height>
```

```
        </build>
    </baldosas>
    <buildings>
        <build>
            <tileX>5</tileX>
            <tileY>1</tileY>
            <source>epsc</source>
            <width>461</width>
            <height>240</height>
            <colision>
                <tileX1>5</tileX1>
                <tileY1>2</tileY1>
                <tileX2>14</tileX2>
                <tileY2>5</tileY2>
            </colision>
        </build>
        <build>
            <tileX>6</tileX>
            <tileY>6</tileY>
            <source>tree</source>
            <width>40</width>
            <height>40</height>
            <colision>
                <tileX1>6</tileX1>
                <tileY1>6</tileY1>
                <tileX2>6</tileX2>
                <tileY2>6</tileY2>
            </colision>
        </build>
    </buildings>
    <buttons>
        <button>
            <id>Biblioteca</id>
            <next>c1f1</next>
            <posX>10</posX>
            <posY>320</posY>
        </button>
    </buttons>
    <games>
        <game>
            <id>Ping Pong</id>
            <link>http://.../FlexPong.swf</link>
            <posX>570</posX>
            <posY>450</posY>
        </game>
    </games>
    <webs>
        <web>
            <id>Web EPSC</id>
            <link>http://epsc.upc.edu/ca/</link>
            <posX>320</posX>
            <posY>10</posY>
```

```
            </web>
        </webs>
</world>
```

Fig. 55 Ejemplo de fichero XML de zona

- *Logo de la zona:* una imagen que sirva de logo de la zona en la vista del mapa general. Puede ser una vista en miniatura o una imagen que tenga relación con la zona.

  El nombre de esta imagen ha de seguir la forma logo-*nombre_del_mundo.* Por ejemplo, si se está añadiendo la zona exterior C1F1 el nombre de la imagen será logo-c1f1, si se añade la zona interior C2F2 que está dentro de la exterior C1F1, el nombre de la imagen será logo-c1f1ic2f2.

  Las dimensiones de la imagen más adecuadas serian 170x150, sin embargo, aunque se suba una imagen de otras dimensiones se podrá visualizar.

- *Las imágenes que forman la zona:* han de adjuntarse todas las imágenes, en formato PNG, que son necesarias para visualizar la zona. Estas imágenes son las que están referenciadas en el archivo XML que describe la zona. Si alguna de estas imágenes no tiene el nombre correcto, o no se adjunta, la zona no se podrá visualizar correctamente.

3.7.2. Juegos, subir la puntuación

En las nuevas zonas que se añaden al mundo virtual pueden existir enlaces a nuevos juegos. Para que estos juegos ayuden a los usuarios a subir sus puntos sociales, se ha diseñado una pequeña API para poder implementar esta funcionalidad.

Para que funcione correctamente, el nuevo juego ha de estar incrustado en una página web o que la aplicación esté accesible en un servidor web. Esta web es la misma que se ha de indicar en el tag link del botón de acceso al juego en el archivo XML de la zona (véase apartado 2.2.1.2).

La API se divide en dos procesos:

1. *Coger el nombre de usuario del jugador.*

   Desde la aplicación del mundo virtual, cuando un usuario clica en un botón de juego, se abre una nueva ventana del navegador redireccionada a la URL del juego.

   En la URL indicada, además se añade un parámetro donde se indica el valor del nombre de usuario. Este parámetro se llama *username*.

Ejemplo:

```
http://dxat.ath.cx:8080/VirtualEPSC/flash/FlexPong.swf?username=noemi
```

Para poder utilizar este parámetro, depende de la tecnología que se haya utilizado en el juego. Si el juego es un archivo Flash, la forma de recoger el valor del parámetro es la siguiente:

```
var username:String = Application.application.parameters.username;
```

2. Cambiar el valor de los puntos sociales del usuario en la base de datos.

Cuando el juego termina, el usuario ha conseguido una serie de puntos. Para poder sumar esos puntos al valor total de los puntos sociales del usuario, se ha de hacer una petición a la base de datos.

Tal y como se ha explicado en el punto 3.6.1.2, la comunicación con la base de datos se hace mediante un Servlet alojado en el servidor web. Por lo tanto, para sumar puntos sociales al usuario, se ha de realizar la petición de tipo subir puntuación al Servlet. Los parámetros necesarios para poder realizar la petición son:

- Type = 2: indica que es una acción de subir puntuación.
- Username: es el nombre de usuario que hay que modificar. Este valor se ha de recoger en el proceso explicado anteriormente.
- Score: el número de puntos conseguidos en el juego.
- Serial: número aleatorio. Puede elegirse cualquier número, por ejemplo la hora en que se hace la petición (Date.getTime()).

Ejemplo:

```
http://ip_servidor_web/VirtualEPSC/services/RequestListener?type=2&username=noemi&score=3&serial=1
```

Desde Flex, se pueden utilizar objetos HTTPService, que permiten hacer peticiones a los Servlets pasándoles diferentes parámetros.

Ejemplo:

```
var httpReq:HTTPService = new HTTPService();
httpReq.url = "http://dxat.ath.cx:8080/VirtualEPSC/services/RequestListener?type=2&username=" + username + "&score=" + score + "&serial=" + serial;
httpReq.send();
```

CAPÍTULO 4. OTRAS TECNOLOGIAS UTILIZADAS

A continuación, se explican aquellas tecnologías que han sido utilizadas como soporte en el desarrollo de este proyecto. Estas tecnologías son: Maven, una herramienta de gestión y construcción de proyectos en Java; Jetty Web Server, el servidor HTTP y contenedor de Servlets utilizado en este proyecto; Subversion, un sistema de control de versiones y repositorio de código fuente.

4.1. MAVEN

En todo proyecto Java siempre existen unas tareas que realizar. La primera suele ser crear una estructura de directorios donde se encuentren los archivos de código fuente, las imágenes, los ficheros de configuración y datos, un directorio para los *.class* o los *.jar*, etc. Además, si el proyecto es grande, es posible que dependa de un gran número de librerías externas

Para facilitar este tipo de tareas aparece Maven [23], una herramienta de software para la gestión y construcción de proyectos en Java basada en el concepto del Project Object Model (POM). Su función es similar a la de Apache Ant, pero el modelo de configuración de construcción es más simple, basado en un formato XML.

Maven es una herramienta de línea de comandos, que con comandos simples crea una estructura de directorios para un proyecto e indicándole cuales son las dependencias de frameworks externos que se necesitan, es capaz de buscarlos en un repositorio, descargarlos e incluirlos en el Build Path del proyecto, ahorrándole toda esa faena al programador.

Maven fue creado por Jason van Zyl, de Sonatype, en 2002. Inicialmente estuvo integrado dentro del proyecto Jakarta, pero actualmente es un proyecto de nivel superior de la Apache Software Foundation.

4.1.1. Uso de Maven

Una vez instalado Maven en el equipo y el plugin de Maven en el IDE Eclipse, es necesario crear el proyecto Wicket. Para ello, se debe introducir el siguiente comando en la línea de comandos de Windows o en un Terminal de Linux o Mac OS X:

```
mvn archetype:create -DarchetypeGroupId=org.apache.wicket -
DarchetypeArtifactId=wicket-archetype-quickstart -
DarchetypeVersion=1.4.8 -DgroupId=org.upc.virtualepsc -
DartifactId=VirtualEPSC
```

Los parámetros utilizados son los siguientes:

- *archetype:create:* es el comando de Maven para crear un proyecto.

- *DarchetypeGroupId:* indica que el proyecto es del tipo Apache Wicket.
- *DarchetypeArtifactId:* inidica que el proyecto sigue una estructura idéntica al del proyecto wicket-archetype-quickstart.
- *DarchetypeVersion:* indica cual es el version de wicket que se desea usar.
- *DgroupId:* es el nombre de paquete inicial, llamado grupo, para todas las clases del proyecto. Todos los proyectos Maven deben pertenecer a un grupo llamado groupId.
- *DartifactId:* es el nombre del proyecto a desarrollar. Todos los proyectos deben tener un nombre para identificarlos llamado artifactId.

Una vez haya acabado de ejecutarse el comando anterior, se ha de añadir el proyecto al IDE específico. Concretamente, en la realización de este proyecto se ha utilizado Eclipse como entorno de desarrollo. Para añadir el proyecto, desde la línea de comandos de Windows o en un Terminal de Linux o Mac OS X, es necesario trasladarse al directorio del proyecto creado anteriormente y ejecutar el siguiente comando:

```
mvn eclipse:eclipse –DdownloadSources=true
```

Una vez ejecutado el comando y añadido a Eclipse, Maven ha creado una estructura de directorios y ficheros como la que se muestra en la figura 56.

```
./VirtualEPSC
+ --- src
|       + --- main
|       |       + --- java
|       |       |       + --- org
|       |       |               + --- upc
|       |       |                       + --- virtualepsc
|       |       |                               VirtualEPSCApplication.java
|       |       + --- resources
|       |               log4j.properties
|       + --- test
|       |       + --- java
|       |               + --- org
|       |                       + --- upc
|       |                               + --- virtualepsc
|       |                                       Start.java
|       + --- webapp
|               + --- WEB-INF
|                       web.xml
+ --- JRE System Library
+ --- Maven Dependencies
        pom.xml
```

Fig. 56 Estructura de directorios creada por Maven

Se crean dos subdirectorios que son *src* y *test*. Dentro de *src* se deben guardar todos los ficheros que sean fuentes y ficheros de configuración o datos del proyecto. En cambio, en *test*, se guardan los ficheros de código fuente de prueba, clases de test de JUnit y ficheros de datos o configuración de pruebas.

Al ser un proyecto Wicket, también aparece el subdirectorio *webapp* donde se suele guardar el fichero *web.xml* que permite configurar cualquier aplicación web en un servidor de aplicaciones como puede ser Apache Tomcat o Jetty. Además, se genera automáticamente el fichero *pom.xml* del proyecto, donde se podrán añadir más dependencias que sean necesarias durante el desarrollo del proyecto.

4.1.2. Project Object Model (POM)

Maven utiliza un Project Object Model o POM para cada proyecto. El POM es un archivo XML, llamado *pom.xml*, que contiene información sobre el proyecto: detalles de configuración utilizados para construirlo, dependencias de otros módulos y componentes externos, etc.

El POM contiene valores por defecto para la mayoría de los proyectos, por ejemplo el directorio de origen (*src/main/java*) o el directorio de fuente de prueba (*src/main/test*).

Algunas de las configuraciones más importantes que se pueden especificar en el POM son las dependencias del proyecto o los complementos. Maven es capaz de descargar automáticamente de la red todas las dependencias o complementos que el proyecto necesita de los repositorios de ficheros jar de Maven. Además, una vez descargados, Maven guarda los archivos jar en un repositorio local en el ordenador para que si se necesita una segunda vez, no sea necesario volver a descargarlo.

Otras informaciones menos importantes que se pueden especificar en el POM son la versión del proyecto, la descripción, los desarrolladores, etc.

4.2. JETTY WEB SERVER

Jetty Web Server [24] es un servidor HTTP y un contenedor de Servlets, ambos basados completamente en Java.

Jetty empezó a desarrollarse en 1995. Desde la versión Jetty-7 en Enero de 2009, el proyecto de Jetty pasó a formar parte de la Eclipse Foundation. Jetty se publica como un proyecto de software libre bajo la licencia Apache 2.0 y la Eclipse Public License 1.0. Actualmente la última versión más estable es la 7, pero existe una versión Jetty-8 experimental.

El desarrollo de Jetty se enfoca en crear un servidor web que sea simple, flexible, eficiente, integrable (en inglés, embedded) y extensible o pluggable.

Además, tiene un tamaño muy pequeño, esta característica lo hace apropiado para ofrecer servicios Web utilizándolo de manera integrada dentro de una aplicación Java, como es el caso de este proyecto.

En la figura 57, se puede observar como es el funcionamiento básico del Servidor Jetty integrado en una aplicación Java.

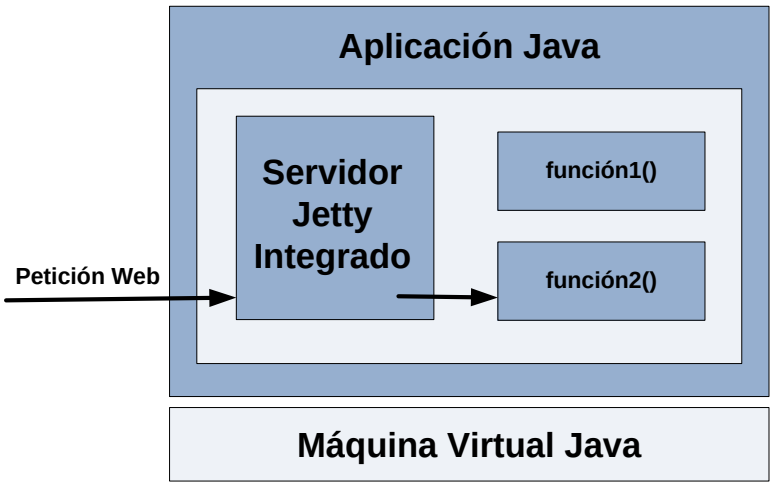

Fig. 57 Servidor Jetty integrado en una aplicación Java

Además, Jetty proporciona:

- Un servidor HTTP tradicional y asíncrono capaz de servir contenido estático y dinámico.
- Un cliente HTTP asíncrono.
- Un contenedor de Servlets.
- Un servidor Web Sockets.
- Soporte para OSGi, JNDI, JMX, AJP.

Para lanzar aplicaciones web que utilizan Jetty existen varias opciones, pero en este proyecto se utilizará el plugin de Maven específico para Jetty. La idea es conseguir que tan sólo con un comando de Maven sea posible arrancar la aplicación web utilizando Jetty como contenedor. Además, utilizando Maven no es necesario descargar ni instalar Jetty, ya que Maven se encarga de ello automáticamente. Para conseguir que se ejecute la aplicación en el puerto 8080, por defecto, hay que configurar el archivo POM de la siguiente manera:

- Indicar cuál es el contextPath, es decir, la carpeta donde se encuentra el proyecto y cuál es su nombre:

```
<groupId>org.upc.vitualepsc</ groupId >
<artifactId>VirtualEPSC</ artifactId >
```

- Indicar que el proyecto se empaquete como war:

```
<packaging>war</packaging>
```

- Configurar Maven para utilizar el plugin de Jetty:

```
<build>
  <plugins>
      <plugin>
            <groupId>org.mortbay.jetty</groupId>
            <artifactId>maven-jetty-plugin</artifactId>
      </plugin>
      <connectors>
            <connector implementation=
                  "org.mortbay.jetty.nio.SelectChannelConnector">
                  <port>9090</port>
             </connector>
      </connectors>
  </plugins>
</build>
```

Finalmente, para arrancar la aplicación web únicamente hay que desplazarse hasta el directorio donde se encuentre el archivo *pom.xml* y utilizar el comando de Maven: *maven jetty:run*.

Hoy en día existen muchos proyectos que utilizan Jetty o están basados en él, como por ejemplo los servidores de aplicación JBoss y Geronimo, el plugin Google Web Toolkit para Eclipse o el soporte Java en el Google App Engine.

4.3. SUBVERSION

Subversion [25], conocido también como svn, es un software de sistema de control de versiones, su funcionamiento se basa en mantener copias de un proyecto en un repositorio pudiendo recuperar cualquier versión de los datos guardados. Actualmente, es un software que se encuentra bajo una licencia Apache/BSD.

Existen varias interfaces que actúan de clientes de Subversion, tanto programas individuales como plugins integrados en entornos de desarrollo. En este proyecto, se han utilizado el programa TortoiseSVN, que es la interfaz más popular para Windows, y el plugin Subclipse que integra el Subversion en el entorno de Eclipse y permite al desarrollador interactuar con el servidor SVN de una manera rápida y sencilla.

Subversion puede acceder al repositorio a través de redes, lo que le permite ser usado por personas que se encuentran en distintos equipos. A cierto nivel, la posibilidad de que varias personas puedan modificar y administrar el mismo

conjunto de datos desde sus respectivas ubicaciones fomenta la colaboración. Como en este proyecto han participado dos personas, este software ha potenciado el desarrollo en grupo.

CAPÍTULO 5. PLAN DEL PROYECTO

Las tareas realizadas en el proyecto se pueden separar en 4 tipos:

- *Estudio previo*: esta tarea se basa en la búsqueda de información, el estudio sobre las tecnologías y los protocolos existentes que se pueden utilizar en el proyecto.

- *Diseño*: engloba el diseño de la arquitectura de la web y de la aplicación. También el diseño grafico de la página web.

- *Implementación*: en este grupo están todas las tareas relacionadas con la programación de la página web y de la aplicación, como también la implementación de la base de datos y la configuración de los servidores necesarios.

- *Documentación*: es la recopilación de información relativa al proyecto y la redacción de la memoria.

5.1. Tiempo de dedicación

En la figura 58, se puede ver como ha sido la evolución de este proyecto a lo largo del tiempo. El período de duración total del proyecto ha sido de aproximadamente unos 6 meses de trabajo. En este período, el trabajo se ha dividido en diferentes etapas: estudio previo, diseño, implementación y documentación.

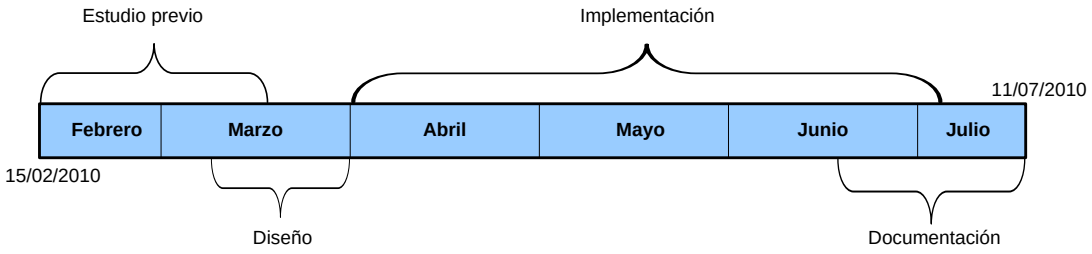

Fig. 58 Esquema del tiempo de dedicación

5.2. Tareas

A continuación se encuentra la tabla de las tareas más importantes realizadas durante las diferentes fases del proyecto. Para cada tarea se ha especificado a qué fase pertenece, una pequeña descripción, el tiempo total que se ha dedicado y el tanto por ciento aproximado que cada componente del grupo ha realizado de esa tarea.

Tabla 5 Tareas del proyecto

Tipo de tarea	Descripción	Luis Miguel	Noemí	Tiempo total
Estudio previo	Búsqueda de información sobre web y juegos 2.0 actuales.	60%	40%	20 horas
Estudio previo	Establecer los objetivos principales.	50%	50%	30 horas
Estudio previo	Búsqueda de información sobre Flex y Action Script.	40%	60%	65 horas
Estudio previo	Instalar las tecnologías principales: Eclipse, Maven, SVN, Flex Builder y el servidor Openfire.	50%	50%	35 horas
Estudio previo.	Búsqueda de información sobre el protocolo XMPP y primeras pruebas en Java.	70%	30%	55 horas
Diseño	Crear el diseño básico de la página web.	50%	50%	40 horas
Diseño	Crear un prototipo de mapa general de la web.	70%	30%	35 horas
Diseño	Diseñar el prototipo de la aplicación del mundo virtual.	30%	70%	40 horas
Implementación	Implementación de la página web.	70%	30%	130 horas
Implementación	Instalación de la base de datos.	80%	20%	10 horas
Implementación	Unión entre la base de datos y la pagina web mediante Java.	65%	35%	50 horas
Implementación	Implementación del registro de usuarios en la web o utilizando OpenId.	35%	65%	60 horas
Implementación	Implementación inicial del mundo virtual.	45%	55%	80 horas
Implementación	Crear interfaz de elección de avatar.	50%	50%	4 horas
Implementación	Implementación del cliente XMPP en Action Script.	35%	65%	20 horas

Implementación	Creación de varios juegos: Trivial, Ping Pong.	10%	90%	12 horas
Implementación	Implementación de servlets para la comunicación entre la aplicación y la base de datos.	40%	60%	30 horas
Implementación	Añadir lista de amigos para cada usuario en la base de datos y creación de la interfaz para agregarlos en el mundo virtual.	35%	65%	10 horas
Implementación	Instalación y configuración del servidor XMPP.	50%	50%	10 horas
Implementación	Ampliación de las funciones de la aplicación del mundo virtual.	30%	70%	50 horas
Documentación	Recopilación de información sobre el contexto del proyecto y las tecnologías utilizadas.	50%	50%	45 horas
Documentación	Redacción de la memoria.	50%	50%	175 horas

TOTAL	1006 horas

Como se ha podido observar en la tabla anterior, la fase más larga ha sido la implementación con un total de 466 horas, ya que se trata de una de las etapas más importantes de este proyecto y es la etapa donde se han realizado el mayor número de tareas. Por otra parte, las fases de estudio previo con 205 horas y diseño con 115 horas, también han ocupado bastante tiempo, e incluso se han solapado entre ellas, ya que muchas de las tareas estaban relacionadas entre ellas. La última fase de documentación ha sumado un total de horas de 220 horas, aunque podrían haber sido más, ya que parte de la información utilizada en esta etapa ya había sido buscada en la etapa de estudio previo.

En la figura 59, se puede ver un gráfico circular que muestra cuales son los porcentajes que se han dedicado para cada una de las cuatro etapas del proyecto.

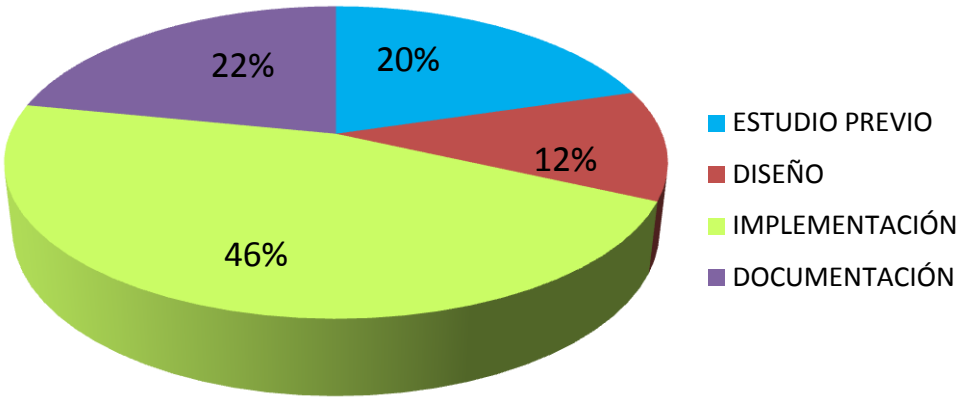

Fig. 59 Gráfico circular del tiempo de dedicación

La fase de estudio previo ha sido muy larga, ya que al utilizar nuevas tecnologías, la curva de aprendizaje para poder comenzar las fases de diseño e implementación ha sido bastante amplia.

CAPÍTULO 6. CONCLUSIONES

En este capítulo, se describen cuáles han sido las conclusiones al finalizar el trabajo. Primero, se mostrarán los objetivos alcanzados respecto al resultado generado. Seguidamente, se hace una pequeña reflexión sobre el impacto medioambiental del proyecto. Finalmente, se encuentran las conclusiones personales junto con algunas posibles mejoras futuras.

6.1. Objetivos alcanzados

Al inicio de este trabajo se definieron las funcionalidades deseadas tanto del sitio web como de la aplicación del mundo virtual. A pesar de que ha existido una curva de aprendizaje bastante amplia, se han conseguido alcanzar la mayor parte de los objetivos.

- Se ha creado una plataforma de un mundo virtual que simula el Campus del Baix Llobregat.
- Para incentivar la participación de los usuarios se han añadido juegos y se ha creado una API mediante la cual, cualquier usuario puede añadir una nueva zona en el mundo virtual. También se ha proporcionado una API para crear nuevos juegos que suban la puntuación de los usuarios.
- Para que la aplicación sea interactiva, se ha añadido un servicio de mensajería instantánea utilizando el protocolo XMPP. De esta forma, todos los usuarios pueden ver que está pasando en cada una de las zonas en tiempo real, y además, los usuarios disponen de un chat público en cada zona y la posibilidad de abrir un chat privado con otros usuarios.
- En la aplicación del mundo virtual se han añadido links a servicios del campus para poder acceder a ellos de forma rápida y sencilla. Además, en la API para añadir nuevas zonas también se proporciona la posibilidad de añadir más links.
- Se ha incluido en varias zonas material lúdico, como por ejemplo juegos que permiten a los usuarios conseguir puntos sociales. Además, mediante las dos APIs proporcionadas, se permite crear nuevos juegos y añadirlos en las nuevas zonas.
- En la web, también se ha añadido una página de noticias para mantener a los usuarios al corriente de nuevas actualizaciones. Las noticias pueden ser añadidas desde la web por usuarios que tengan rol de administradores o desarrolladores.
- Se ha utilizado OpenID para facilitar el registro y el acceso a la web utilizando una cuenta de Google previamente creada.
- El sistema de almacenamiento que se ha utilizado es un sistema escalable y distribuido que permite alojar grandes cantidades de información.

Como se puede observar, se han cumplido la totalidad de los objetivos propuestos al inicio del proyecto. Ha sido de vital importancia para conseguir estos objetivos el uso de nuevas tecnologías, como Apache Wicket, para el

desarrollo de la web; OpenID4Java, para agilizar el proceso de registro y autenticación de usuario; Adobe Flex, en la aplicación del mundo virtual; Apache Cassandra, como sistema de almacenamiento de datos y el protocolo XMPP, como servicio de mensajería instantánea.

6.2. Impacto medioambiental

A primera vista, parece que este trabajo no tiene ningún impacto sobre el medio ambiente. Sin embargo, sí que existe una pequeña influencia.

El hecho de que la aplicación del mundo virtual emule el Campus del Baix Llobregat, permite que futuros estudiantes de la universidad no tengan la necesidad de asistir físicamente para conocerlo. De esta manera, se consigue reducir tanto los desplazamientos en vehículos privados como en públicos y, en consecuencia, se reducen las emisiones de CO_2 a la atmosfera.

En las futuras ampliaciones de la aplicación, se pretende que los usuarios puedan compartir apuntes sobre las diferentes asignaturas en formato digital (documentos de Word o PDF). Con esta funcionalidad, se consigue reducir el número de fotocopias que realizan los alumnos para intercambiar apuntes. Así mismo, se reduce el consumo de papel, evitando que se talen más árboles; además de reducir los niveles de tinta, que en la mayoría de los casos suelen ser tóxicas y perjudican gravemente el medio ambiente.

El hecho de que tanto la aplicación, como el portal web o el resto de servicios ofrecidos no requieren de máquinas con demasiados recursos, hace que pueda utilizarse una técnica que se denomina *virtualización*. El término virtualización hace referencia a que en una máquina física se puedan emular N máquinas virtuales, donde cada una de ellas trabaja de manera independiente. De esta manera, se podrían alojar en cada una de estas máquinas virtuales los diferentes servicios del proyecto, como son el servidor XMPP, el servidor de aplicaciones, etc.; sin necesidad de utilizar N máquinas reales, con lo que se permite reducir considerablemente el consumo eléctrico del sistema.

6.3. Conclusiones personales

La realización de este trabajo, nos ha aportado una serie de conocimientos que no teníamos, además de aportarnos una mayor experiencia en la elaboración de proyectos en grupo.

A lo largo de la carrera, habíamos trabajado con diferentes tecnologías que se suelen utilizar en aplicaciones telemáticas, pero todas ellas estaban ya suficientemente implantadas o existía suficiente información para entender su uso. En el caso de las tecnologías utilizadas en este trabajo, hemos tenido algunos problemas para interactuar con algunas tecnologías como Apache Cassandra, por el simple hecho de ser una tecnología muy novedosa y en desarrollo. Sin embargo, otras tecnologías como es el caso de Apache Wicket,

nos ha facilitado el desarrollo del portal web, de una manera sencilla, ya que nunca habíamos desarrollado un portal de tal complejidad.

Gracias a que gran parte del proyecto está desarrollado en Java, un lenguaje de programación que ya conocíamos y habíamos trabajado con él a lo largo de la carrera, se han podido integrar todas las tecnologías de manera rápida y sencilla.

El hecho de desarrollar la aplicación del mundo virtual en Adobe Flex, un lenguaje de programación desconocido para nosotros, ha supuesto un hándicap que hemos sabido superar exitosamente.

En definitiva, ha sido un trabajo muy interesante ya que engloba muchos conceptos que se utilizan constantemente en la mayoría de aplicaciones telemáticas. Además, al tratarse de un proyecto conjunto nos ha permitido enfrentarnos y superar los problemas que surgen al trabajar en grupo.

6.4. Futuras mejoras

Aunque todos los objetivos iniciales del proyecto han sido alcanzados con éxito, existen una serie de funcionalidades bastante útiles e interesantes que se podrían implantar en un futuro:

- Creación de un editor de mapas gráficos que permita desarrollar una zona del mundo virtual de una manera sencilla e intuitiva.
- Creación de un servicio que permita a los alumnos compartir apuntes de forma digital.
- Mejora de la interfaz de elección de avatares, pudiendo personalizar los rasgos de la cara, la ropa, accesorios, etc.
- Desarrollo de una aplicación que permita una comunicación directa entre usuarios, como pueden ser videoconferencias, ya sean entre pares de usuarios o en grupos.
- Ofrecer un mecanismo basado en HTTP/SOAP u otros protocolos propietarios para que los usuarios puedan colaborar en la generación de contenidos.
- Establecer un servicio de posts de perfiles o muros en la web similar al de otras redes sociales existentes: Facebook, Twitter, Tuenti, etc.
- Permitir la compartición de media entre usuarios en la web: creación de álbumes de fotos, subida de vídeos, publicación de enlaces externos, etc.
- Posible adaptación del proyecto a otras nuevas técnicas de desarrollo web, como por ejemplo HTML5.

BIBLIOGRAFÍA

[1] **La explosión de la Burbuja puntocom:** Articulo online que describe las características del fenómeno de la burbuja de los sitios ".com". (En línea). [Última Consulta: 5 de julio de 2010].
URL: http://knol.google.com/k/la-explosión-de-la-burbuja-punto-com

[2] **Second Life:** Web oficial del famoso juego Second Life, donde se simula un mundo virtual en 3D. (En línea). [Última Consulta: 1 de junio de 2010].
URL: http://secondlife.com/

[3] **UMH Virtual:** Web oficial de la Universidad Miguel Hernández de Elche sobre la creación de un mundo virtual en 3D simulando todos sus campus. (En línea). [Última Consulta: 5 de junio de 2010].
URL: http://virtual.umh.es/

[4] **Web 2.0 SUMMIT:** Web oficial sobre la conferencia Web 2.0 SUMMIT que trata sobre el concepto web 2.0. (En línea). [Última Consulta: 15 de junio de 2010].
URL: http://www.web2summit.com

[5] **Apache Wicket:** Web oficial sobre el proyecto Apache Wicket. (En línea). [Última Consulta: 1 de julio de 2010].
URL: http://wicket.apache.org/

[6] Gurumurthy, K., *Pro Wicket*, Anglin, S., Nueva York, 2006.

[7] **Apache Cassandra:** Web oficial sobre el proyecto Apache Cassandra. (En línea). [Última Consulta: 19 de junio de 2010].
URL: http://cassandra.apache.org/

[8] **No-SQL:** Web con información referente a bases de datos que siguen la estructura No-SQL. (En línea). [Última Consulta: 30 de junio de 2010].
URL: http://nosql-database.org/

[9] **Cassandra – A Decentralized Structured Storage System:** Artículo online que describe un estudio detallado sobre Cassandra realizado por dos miembros del equipo de desarrollo de Facebook. (En línea). [Última Consulta: 1 de julio de 2010].
URL: http://www.cs.cornell.edu/projects/ladis2009/papers/lakshman-ladis2009.pdf

[10] **Cassandra – Structured Storage System over a P2P Network:** Presentación online que describe un estudio detallado sobre Cassandra realizado por dos miembros del equipo de desarrollo de Facebook. (En línea). [Última Consulta: 1 de julio de 2010].
URL: http://static.last.fm/johan/nosql-20090611/cassandra_nosql.pdf

[11] **Apache Thrift:** Web oficial sobre el proyecto Apache Thrift. (En línea). [Última Consulta: 19 de junio de 2010].
URL: http://incubator.apache.org/thrift/

[12] **OpenID:** Web oficial sobre el proyecto OpenID acerca de identidades universales. (En línea). [Última Consulta: 7 de julio de 2010].
URL: http://openid.net/

[13] **OpenID4Java:** Web oficial sobre el framework de Java OpenID4Java que permite interactuar con OpenID. (En línea). [Última Consulta: 29 de junio de 2010].
URL: http://code.google.com/p/openid4java/

[14] **OpenID for Google Accounts:** Web oficial de Google que específica como debe realizarse el proceso de autenticación de sus cuentas mediante OpenID. (En línea). [Última Consulta: 4 de julio de 2010].
URL: http://code.google.com/intl/es-ES/apis/accounts/docs/OpenID.html

[15] **XMPP:** Web oficial de la XMPP Standards Foundation creadora del protocolo XMPP. (En línea). [Última Consulta: 26 de junio de 2010].
URL: http://xmpp.org/

[16] **Openfire:** Web oficial de la Ignite Realtime sobre el proyecto Openfire como servidor XMPP. (En línea). [Última Consulta: 19 de junio de 2010].
URL: http://www.igniterealtime.org/projects/openfire/

[17] **API Smack:** Web oficial de la Ignite Realtime sobre la librería cliente de XMPP para Java. (En línea). [Última Consulta: 22 de junio de 2010].
URL:http://www.igniterealtime.org/projects/smack/

[18] **API XIFF:** Web oficial de la Ignite Realtime sobre la librería cliente de XMPP para Flash. (En línea). [Última Consulta: 30 de junio de 2010].
URL: http://www.igniterealtime.org/projects/xiff/index.jsp

[19] **Adobe Flex:** Web oficial sobre el proyecto Adobe Flex. (En línea). [Última Consulta: 5 de julio de 2010].
URL: http://www.adobe.com/es/products/flex/

[20] **Action Script:** Web para desarrolladores de Action Script con gran cantidad de tutoriales y ejemplos. (En línea). [Última Consulta: 8 de julio de 2010].
URL: http://www.actionscript.org/

[21] **Tweening Platforms de GreenSock:** Web oficial de GreenSock sobre el framework que permite el movimiento de objetos mediante el algoritmo A* en Flash. (En línea). [Última Consulta: 11 de junio de 2010].
URL: https://www.greensock.com/v11/

[22] **Algoritmo A*:** Web oficial del Laboratorio de Infraestructuras de Datos Especiales de la Universidad de Valladolid donde se explican los conceptos básicos del algoritmo de búsqueda A*. (En línea). [Última Consulta: 2 de julio de 2010].
URL: http://idelab.uva.es/algoritmo

[23] **Apache Maven:** Web oficial sobre el proyecto Apache Maven. (En línea). [Última Consulta: 1 de julio de 2010].
URL: http://maven.apache.org/

[24] **Jetty Web Server:** Web oficial sobre el proyecto Jetty de la Codehaus Foundation. (En línea). [Última Consulta: 11 de junio de 2010].
URL: http://jetty.codehaus.org/jetty/

[25] **Subversion:** Web oficial del proyecto Apache Subversion. (En línea). [Última Consulta: 10 de junio de 2010].
URL: http://subversion.apache.org/

Algunos de los conceptos definidos, se han obtenido de: Wikipedia – La enciclopedia libre (En línea).
URL: http://es.wikipedia.org/wiki/Wikipedia:Portada

ACRÓNIMOS, SIGLAS Y DEFINICIONES

ATOM (*Atom Syndication Format*) – Lenguaje XML que se utiliza para realizar web feeds o medios de redifusión de contenidos web.

AJAX (*Asynchronous JavaScript And XML*) – Técnica de desarrollo web que permite modificar la información de una página web sin tener que recargarla completamente, agilizando de esta forma la interacción con el usuario.

AJP (*Apache JServ Protocol*) – Protocolo binario que puede llevar a cabo las peticiones entrantes desde un servidor web a través de un servidor de aplicaciones que se encuentra detrás del servidor web.

API (*Application Programming Interface*, Interfaz de Programación de Aplicaciones) – Conjunto de especificaciones para comunicarse con una aplicación, normalmente para obtener información y utilizarla en otros servicios. Ejemplos: *Amazon Web Services, Flickr Services, Google AJAX API.*

CSS (*Cascading Style Sheets, Hojas de Estilo en Cascada*) – Lenguaje para definir la presentación de las páginas web, de modo que su aspecto quede separado del contenido en sí.

DNS (*Domain Name System*) – Sistema de nomenclatura jerárquica para computadoras, servicios o cualquier recurso conectado al internet o a una red privada. Su función más importante, es traducir (resolver) nombres inteligibles para los humanos en identificadores binarios asociados con los equipos conectados a la red, esto con el propósito de poder localizar y direccionar estos equipos mundialmente.

Framework – Estructura conceptual y tecnológica de soporte definida, normalmente con artefactos o módulos de software concretos, con base en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros programas para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

HTML (*HyperText Markup Language*) – Lenguaje basado en etiquetas predominante para la elaboración de páginas web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes.

HTTP (*Hypertext Transfer Protocol*) – Protocolo de la capa de Aplicación utilizado en cada transacción de la World Wide Web.

Java Swing – Biblioteca gráfica de Java que permite crear interfaz gráfica de usuario mediante widgets como cajas de texto, botones, desplegables y tablas.

Java Web Start – Software desarrollado por Sun que permite descargar y ejecutar aplicaciones Java desde la Web.

JCC (*JavaScript Client Communication*) – Técnicas de programación que, utilizando objetos JSI en el navegador (en el lado cliente y no en el servidor), facilitan la integración en la misma página Web de aplicaciones y servicios a priori independientes.

JMX (*Java Management Extensions*) – Tecnología que define una arquitectura de gestión, la API, los patrones de diseño, y los servicios para la monitorización/administración de aplicaciones basadas en Java

JNDI (*Java Naming and Directory Interface*) – API de Java para servicios de directorios. Permite a los clientes descubrir y buscar objetos y nombres a través de un nombre y, como todas las APIs de Java que hacen de interfaz con sistemas host, es independiente de la implementación subyacente.

JUnit – Framework que permite realizar la ejecución de clases Java de manera controlada, para poder evaluar si el funcionamiento de cada uno de los métodos de la clase se comporta como se espera

Mashup o Aplicación híbrida – Sitio web o aplicación web que usa contenido de otras aplicaciones Web para crear un nuevo contenido completo, consumiendo servicios directamente, siempre a través de protocolo http.

OSGi – Sistema de módulos y plataforma de servicios para Java que implementa un modelo de componentes completo y dinámico, algo que no existía en Java o entornos VM. Las aplicaciones o los componentes (que viene en forma de paquetes para el despliegue) pueden ser instalados de forma remota, iniciado, parado, actualizado y desinstalado sin necesidad de reiniciar.

REST (*Representational State Transfer*) – Técnica de arquitectura software para sistemas hipermedia distribuidos como la red Internet.

RSS (*Really Simple Syndication*) – Formato estándar para la sindicación de contenidos a los que un usuario cualquiera puede suscribirse mediante un programa «agregador de feeds o canales».

Servlet – Objetos que funcionan dentro del contexto de un contenedor de servlets o aplicaciones. Su función es generar páginas web de forma dinámica a partir de los parámetros de la petición que envíe el navegador web.

SOAP (*Simple Object Access Protocol*) – Protocolo estándar muy utilizado en los servicios Web, que define cómo dos objetos en diferentes procesos pueden comunicarse por medio del intercambio de datos XML.

Socket – Concepto abstracto por el cual dos programas (posiblemente situados en máquinas distintas) pueden intercambiar cualquier flujo de datos, en general de manera fiable y ordenada. El socket se define por medio de una dirección IP, un protocolo de transporte y un número de puerto.

TCP (*Transmission Control Protocol*) – Protocolo fundamental de Internet de la capa de transporte cuya función es el transporte de datos bidireccional y de manera segura.

URL (*Uniform Resource Locator*) – Secuencia de caracteres, de acuerdo a un formato modélico y estándar, que se usa para nombrar recursos en Internet para su localización o identificación, como por ejemplo documentos textuales, imágenes, videos, presentaciones digitales, etc.

WebDAV (*Web-based Distributed Authoring and Versioning*) – Conjunto de extensiones del protocolo HTTP que permiten a los usuarios de ordenadores, editar y gestionar ficheros de manera colaborativa en servidores Web remotos.

Web Sockets – Tecnología que proporciona canales de comunicación bidireccionales sobre un solo socket TCP diseñado para ser implementado en navegadores web y servidores web.

World Wide Web – Sistema de documentos de hipertexto enlazados y accesibles a través de Internet. Con un navegador web, un usuario visualiza sitios web compuestos de páginas web que pueden contener texto, imágenes, videos u otros contenidos multimedia, y navega a través de ellas usando enlaces.

XHTML (*eXtensible Hypertext Markup Language*) – Versión XML más avanzada del languaje HTML que se utiliza para la creación y visualización de páginas web.

XML (*eXtensible Markup Language*) – Metalenguaje de uso general que sirve para definir otros lenguajes de programación o formatos de intercambio de información según diversas necesidades.

XRDS (eXtensible Resource Descriptor Sequence) – Formato XML utilizado para el descubrimiento de metadatos sobre un recurso, en particular descubrimiento de servicios asociados con un recurso.

XRI (Extensible Resource Identifier) – Nuevo sistema de identificación en Internet, diseñado específicamente para identidades digitales de dominio cruzado.

XUL (*XML-based User-interface Language)* – Lenguaje basado en XML para la interfaz de usuario desarrollado por los creadores de Mozilla Firefox.

ANEXOS

A. Fichero de configuración del mapa general

A continuación, se muestra un ejemplo del fichero maps.xml:

```
<?xml version="1.0" encoding="UTF-8" ?>
<map>
 <tilesX>4</tilesX>
 <tilesY>3</tilesY>

 <world>
      <tileX>1</tileX>
      <tileY>2</tileY>
      <name>Lago</name>
      <source>logo-c1f2.png</source>
      <tilesX>0</tilesX>
      <tilesY>0</tilesY>
 </world>

 <world>
      <tileX>2</tileX>
      <tileY>1</tileY>
      <name>entrada EPSC</name>
      <source>logo-c2f1.png</source>
      <tilesX>2</tilesX>
      <tilesY>2</tilesY>
      <worldI>
            <tileX>0</tileX>
            <tileY>1</tileY>
            <name>Lab 333</name>
            <source>logo-c2f1ic0f1.png</source>
      </worldI>
      <worldI>
            <tileX>1</tileX>
            <tileY>0</tileY>
            <name>sala de actos</name>
            <source>logo-c2f1ic1f0.png</source>
      </worldI>
      <worldI>
            <tileX>0</tileX>
            <tileY>0</tileY>
            <name>EPSC</name>
            <source>logo-c2f1ic0f0.png</source>
      </worldI>
 </world>

 <world>
      <tileX>0</tileX>
       <tileY>0</tileY>
```

```
        <name>Estacion de Renfe</name>
        <source>logo-c0f0.png</source>
        <tilesX>1</tilesX>
         <tilesY>1</tilesY>
  </world>

  <world>
        <tileX>1</tileX>
        <tileY>1</tileY>
        <name>entrada biblioteca</name>
        <source>logo-c1f1.png</source>
        <tilesX>2</tilesX>
        <tilesY>2</tilesY>
        <worldI>
              <tileX>1</tileX>
              <tileY>1</tileY>
              <name>Cafeteria</name>
              <source>logo-c1f1ic1f1.png</source>
         </worldI>
        <worldI>
              <tileX>0</tileX>
              <tileY>1</tileY>
              <name>1º planta biblioteca</name>
              <source>logo-c1f1ic0f1.png</source>
        </worldI>
        <worldI>
               <tileX>0</tileX>
              <tileY>0</tileY>
              <name>2º planta biblioteca</name>
              <source>logo-c1f1ic0f0.png</source>
        </worldI>
  </world>
</map>
```

Fig. 60 Ejemplo de fichero maps.xml

B. Apache Wicket

Apache Wicket fue diseñado por Jonathan Locke y se empezó a implementar en Abril de 2004. La primera versión se hizo pública en Junio de 2005, pero no fue hasta Junio de 2007, cuando se integró en un proyecto de Apache, con el nombre de Apache Wicket.

El equipo de Apache Wicket comenzó a desarrollar el framework teniendo dos premisas en la cabeza: separar claramente vista y lógica, y programar en Java todo el comportamiento.

Esto supone un cambio bastante importante a la hora de desarrollar aplicaciones web respecto a JSP, Struts o incluso JSF. En todos estos frameworks se confunden tags de vista con tags de lógica. Sin embargo, la idea de Wicket es coger el HTML, añadir un atributo wicket:id a cada tag que deba tener comportamiento dinámico, y gestionar toda la lógica de la página en servidor.

En Wicket, cada página HTML va acompañada de una clase Java que controla su comportamiento. De esta forma, se permite a desarrolladores noveles o aquellos desarrolladores que no están muy familiarizados con el desarrollo web, realizar aplicaciones web muy potentes sin tener que programar HTML.

Los frameworks tradicionales basados en MVC (Modelo-Vista-Controlador) trabajan en términos de conjunto de peticiones y de páginas completas. En cada ciclo de peticiones, la petición entrante es mapeada en un método en el objeto *controlador*, que en ese instante, genera la respuesta en su totalidad, por lo general, extrayendo información de un *modelo* para poblar una *vista* en HTML. Esto hace que el control de flujo de la aplicación sea simple y claro, pero que la reutilización de código en el controlador sea difícil.

Por otro lado, Wicket es un framework que sigue el modelo de GUI (Graphical User Interface o Interfaz Gráfica de Usuario) con estados muy similar al funcionamiento del framework de Java Swing. Las aplicaciones Wicket son árboles de componentes que usan *listeners* para reaccionar ante peticiones HTTP a través de enlaces o formularios de la misma manera que los componentes de Swing reaccionan a los eventos del ratón o de las teclas.

Wicket utiliza XHTML plano (Extensible Hypertext Markup Language), un lenguaje de programación similar al HTML pero basado en la sintaxis del XML. XHTML es usado para realizar las plantillas de las páginas web y de esta manera se realiza una clara separación de la presentación respecto de la lógica de negocio y permite que estas plantillas puedan ser editadas con los programas convencionales de diseño web basados en la técnica de WYSIWYG (What You See Is What You Get – Lo que ves es lo que obtienes). Estas aplicaciones permiten editar las páginas web viendo directamente el resultado final sin entrar en el propio código. Un buen ejemplo de estas aplicaciones pueden ser NVU, Microsoft FrontPage o Artisteer.

Cada componente Wicket está ligado a un elemento con nombre en el XHTML y se convierte en responsable de la presentación de ese elemento en el resultado final. El concepto de *página o page* en Wicket es simplemente un contenedor de componentes Wicket de alto nivel y que se encuentra emparejado con su plantilla XHTML. Por tanto, en toda aplicación Wicket debe haber una página con su homólogo en XHTML, por ejemplo, si existe una página principal llamada Home, debe existir una clase llamada Home.java, que es la clase que hace referencia a la página Wicket, y Home.html, que es su plantilla XHTML.

Como ya se ha mencionado anteriormente, Wicket no mezcla código Java con el markup HTML y no añade ninguna sintaxis especial a sus archivos .html (como JSF o ASP). Los mundos de XHTML y Java son paralelos y se unen sólo por Wicket ids (wicket:id), que son los atributos en XHTML y propiedades de los componentes en Java. Por tanto, se puede modificar por programación Java cualquier atributo de una etiqueta XHTML. De esta manera, programadores y diseñadores pueden trabajar de forma independiente en gran medida, y sin depender de ninguna herramienta especial.

Partes reutilizables de las páginas Wicket se pueden abstraer en componentes llamados *panels o paneles* que pueden ser incluidos en otras páginas o en otros paneles con un tag especial. Además de los paneles, existe otra técnica que permite reutilizar código, y se conoce como la herencia de páginas o herencia de markups. Mediante esta técnica, una página puede heredar componentes y paneles de otra página web de la cual extiende.

Cada componente está respaldado por su propio *modelo* que representa el estado del componente. El propio framework no tiene conocimiento de cómo los componentes interactúan con sus modelos, que son tratados como objetos opacos automáticamente serializados y persistentes entre peticiones. Estos modelos pueden ser simplemente POJOs (Plain Old Java Object), clases Java muy simples que solo contienen métodos getters y setters, que devuelven o asignan valores a los atributos de esa clase. Estos modelos son realmente útiles en el caso de recoger datos de un formulario, ya que simplemente con un POJO somos capaces de recoger el valor de los campos del formulario.

Además Wicket incorpora AJAX (Asynchronous JavaScript And XML), una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios, mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, lo que significa aumentar la interactividad, velocidad y usabilidad en las aplicaciones. Con esta tecnología es realmente fácil incluir los validadores de campos, como puede ser un validador de correo electrónico que comprueba que ese campo tenga la forma *user@example.com*.

Wicket es seguro por defecto ya que la URL no expone ninguna información sensible. Existen planes de futuro para la próxima versión de Wicket para añadir soporte de codificación URL de sitios web altamente seguro.

En cuanto al tema de sesiones, Wicket elimina la necesidad de gestionar los atributos de HttpSession a mano. Permite crear un tipo de sesión personalizado, además de poder extender de sesiones ya creadas como es el caso de la AuthenticatedWebSession, que además proporciona métodos muy útiles, como por ejemplo, métodos para saber si un usuario se ha autenticado, o asignar roles a usuarios y restringir el acceso de usuarios a una determinada página de la web dependiendo el rol.

C. Apache Cassandra

El proyecto Cassandra es open-source (código abierto), fue inicialmente desarrollado por Facebook en 2008 y, actualmente, se trata de un proyecto de alto nivel de la Apache Software Foundation. Jeff Hammerbacher, quien dirigió el equipo de datos de Facebook que inicio el desarrollo de Cassandra, define el proyecto como un modelo de datos BigTable que se ejecuta en una infraestructura distribuida basada en la de Amazon's Dynamo:

- BigTable es una base de datos comprimida y de alto rendimiento. Es propiedad de Google, actualmente no se distribuye, sólo se utiliza en Google File System (GFS) y otros proyectos de la compañía. Sin embargo, Google ofrece acceso a esta base de datos mediante Google App Engine.

  El sistema de almacenamiento BigTable se aparta de las convenciones típicas de un número fijo de columnas, está diseñado para albergar múltiples dimensiones.

  Al igual que BigTable, Cassandra proporciona un modelo de datos basado en ColumnFamily. Este modelo es mucho más rico que el simple clave / valor.

- Amazon's Dynamo es una base de datos que se especifica como una gran tabla de hash distribuida (DHT). Es propiedad de la empresa Amazon y tiene el código cerrado, por lo que su uso es exclusivo de la empresa. En octubre de 2007 se publicó un paper[1] con el diseño y la especificación, a grandes rasgos, de Dynamo, rompiendo con conceptos como la consistencia o el modelo relacional. Su objetivo se definió claramente: escalabilidad y disponibilidad.

  Al igual que el Dynamo, Cassandra pretende ser consistente, escalable y con una alta disponibilidad.

Este sistema es una solución NoSQL, un movimiento a favor de los almacenes de datos no relacionales que rompen con una larga historia de bases de datos relacionales. El término NoSQL fue usado por primera vez en 1998 como el nombre de una base de datos open-source semi-relacional que no usaba interfaz SQL. El término fue reintroducido a principios del 2009 por un empleado de Rackspace, Eric Evans, cuando Johan Oskarsson, miembro de Last.fm, propuso organizar un evento para discutir sobre bases de datos open-source distribuidas.

El nombre NoSQL fue, por tanto, un intento de describir la aparición de un gran número de almacenes de datos distribuidos y no relacionales que cumplían las garantías ACID. El término ACID proviene del acrónimo de Atomicity, Consistency, Isolation and Durability, que en español sería:

---

[1]Dynamo: Amazon's Highly Available Key-value Store, disponible en: http://www.allthingsdistributed.com/2007/10/amazons_dynamo.html

- Atomicidad: asegurar que la operación se ha realizado o no, y por tanto, ante un fallo del sistema no puede quedarse en medio de un proceso.

- Consistencia: asegurar que sólo se ejecutan aquellas operaciones que no van a romper las reglas y directrices de integridad de la base de datos.

- Aislamiento: asegurar que una operación no puede afectar a otras, asegurando que la realización de dos transacciones sobre la misma información son independientes y no generan ningún tipo de error.

- Durabilidad: asegurar que una vez realizada la operación, ésta persistirá y no se podrá deshacer aunque falle el sistema.

Con el paso del tiempo, el movimiento NoSQL ha ido definiéndose poco a poco pero cada vez mejor, y de hecho, han surgido implementaciones muy potentes basadas en este esquema como pueden ser: Google's BigTable, Amazon's Dynamo, HBase o el propio Cassandra.

Los almacenes de datos NoSQL, como Cassandra, requieren esquemas fijos de tablas, normalmente prohíben operaciones de uniones y típicamente están escalados de manera horizontal.

Otro de los puntos fuertes de Cassandra, es la posibilidad de crear un clúster de nodos para crear un sistema de almacenamiento de alta disponibilidad y sin ningún punto de fallos. Las principales características del clúster de Cassandra son:

- Tolerancia a fallos: los datos son replicados en múltiples nodos miembros del clúster. Por tanto, en caso de que un nodo caiga, los datos que almacena éste no se pierden ya que otros nodos del clúster almacenan una copia de estos datos. Además el nodo que cae es remplazado de forma muy rápida, y por tanto, no se nota su ausencia al no haber prácticamente tiempos de inactividad.

- Descentralización: toda su estructura está basada en un clúster de nodos que son totalmente idénticos entre ellos, es decir, no hay una jerarquía de nodos. Al no tener un punto central donde se guarde la información, no existe un punto único de fallos y, por tanto, tampoco se producen cuellos de botella.

- Flexibilidad: por cada nodo que se añade al clúster, los throughput de lectura y escritura o volúmenes de información que fluyen a través del clúster se ven incrementados de forma lineal.

- Alta disponibilidad: gracias a las características anteriores, la información está siempre disponible. Por eso es un sistema adecuado para aplicaciones que no pueden permitirse perder datos.

Actualmente, Cassandra se está utilizando en Digg, Facebook, Twitter, Reddit, Rackspace, Cloudkick, Cisco, SimpleGeo, Ooyala, OpenX, y más empresas que necesitan una gran base de datos activa.

C.1. Modelo de datos

Como se ha mencionado anteriormente, Cassandra no soporta un modelo de datos relacional completo, en su lugar, proporciona clientes con un modelo de datos simple que soporta un control dinámico sobre la disposición de los datos y el formato.

Dentro del modelo de datos que define Cassandra se pueden encontrar cinco tipos de estructuras de datos:

- Column (Columna): se define como una tupla (triplete) que contiene un nombre, un valor y un timestamp. Esta es la estructura o contenedor datos mínimo que existe en Cassandra.

  A continuación, se puede ver una representación en Java del concepto de Column:

  ```
  public class Column {
    Byte[] name;
    Byte[] value;
    Long timestamp;
  }
  ```

  Se puede representar mediante una clase de datos, llamada Column, que contiene tres atributos:

  - El atributo *name,* que es un array de bytes que contiene el nombre o clave mediante la cual es posible indexar el dato.
  - El atributo *value,* es un array de bytes que contiene el valor.
  - El atributo *timestamp,* es de tipo Long e indica el instante de tiempo en el que fue almacenado el dato.

- SuperColumn (SuperColumna): es una tupla con un nombre (*name*) y un valor (*value*), pero no tiene timestamp a diferencia de la Column. En este caso, el campo *value* no es un valor binario sino que es un mapa que contiene combinaciones de clave/columna. Lo importante en este caso, es que la clave tiene el mismo valor que el nombre de la columna a la que se refiere. Se puede decir que una SuperColumn es un contenedor de una o más Columns.

  A continuación, se puede observar una representación en Java del concepto de SuperColumn:

  ```
  public class SuperColumn {
    Byte[] name;
    // The key is equal to the name of the Column
    Map<Byte[] /* key */, Column> value = null;
  }
  ```

Se puede representar mediante una clase de datos que se llama SuperColumn que contiene dos atributos:

- El atributo *name,* que es un array de bytes que contiene el nombre o clave mediante la cual se podrá indexar la SuperColumn.
- El mapa de Columns, formado por una pareja de clave (array de bytes) y Column.

- ColumnFamily (Familia de Columnas): es una estructura que puede tener un número infinito de filas. Es un concepto similar al de Tabla de las bases de datos relacionales. Toda ColumnFamily cuenta con un nombre, un mapa con una clave y un valor. El valor del mapa es otro mapa que contiene Columns. El mapa con las Columns sigue las mismas reglas que la SuperColumn, donde la clave tiene el mismo valor que el nombre de la Column que indexa.

  A continuación, se puede ver una representación en Java del concepto de ColumnFamily:

```
public class ColumnFamily {
   Byte[] name;
  // The key is a user generated key
  Map<Byte[] /* key */,
    // The key is equal to the name of the Column.
  Map<Byte[] /* key */, Column>> value = null;
}
```

  Se puede representar mediante una clase de datos que se llama ColumnFamily que contiene dos atributos:

  - El atributo *name* que es un array de bytes que contiene el nombre o clave mediante la cual se podrá indexar la ColumnFamily.
  - El mapa de mapas que contienen Columns, formado por una pareja de clave (array de bytes) y un mapa.

- SuperColumnFamily (Familia de SuperColumnas): es una de las estructuras de datos más grandes que existen en el modelo de datos de Cassandra. Añade una nueva dimensión al concepto de SuperColumn. Por tanto, en lugar de tener Columns tenemos un mapa de SuperColumns. De la misma manera, la clave del mapa que contiene las SuperColumns debe ser el mismo que el nombre de la SuperColumn a la que hace referencia.

  A continuación, se puede observar una representación en Java del concepto de SuperColumnFamily:

```
public class SuperColumnFamily {
  Byte[] name;
  // The key is a user generated key
  Map<Byte[] /*key */,
    // The key is equal to the name of the SuperColumn.
  Map<Byte[] /* key */,
```

```
        SuperColumn>> value = null;
}
```

Se puede representar mediante una clase de datos que se llama SuperColumnFamily y contiene dos atributos:

- El atributo *name* que es un array de bytes que contiene la clave mediante la cual se podrá indexar la SuperColumnFamily.
- El mapa de mapas que contienen SuperColumns, formado por una pareja de clave (array de bytes) y un mapa.

- Keyspace (Espacio de Claves): es la estructura más amplia del modelo de datos de Cassandra y, por tanto, almacena ColumnFamilies y SuperColumnFamilies. Es un concepto muy similar al de Esquema de las bases de datos relacionales.

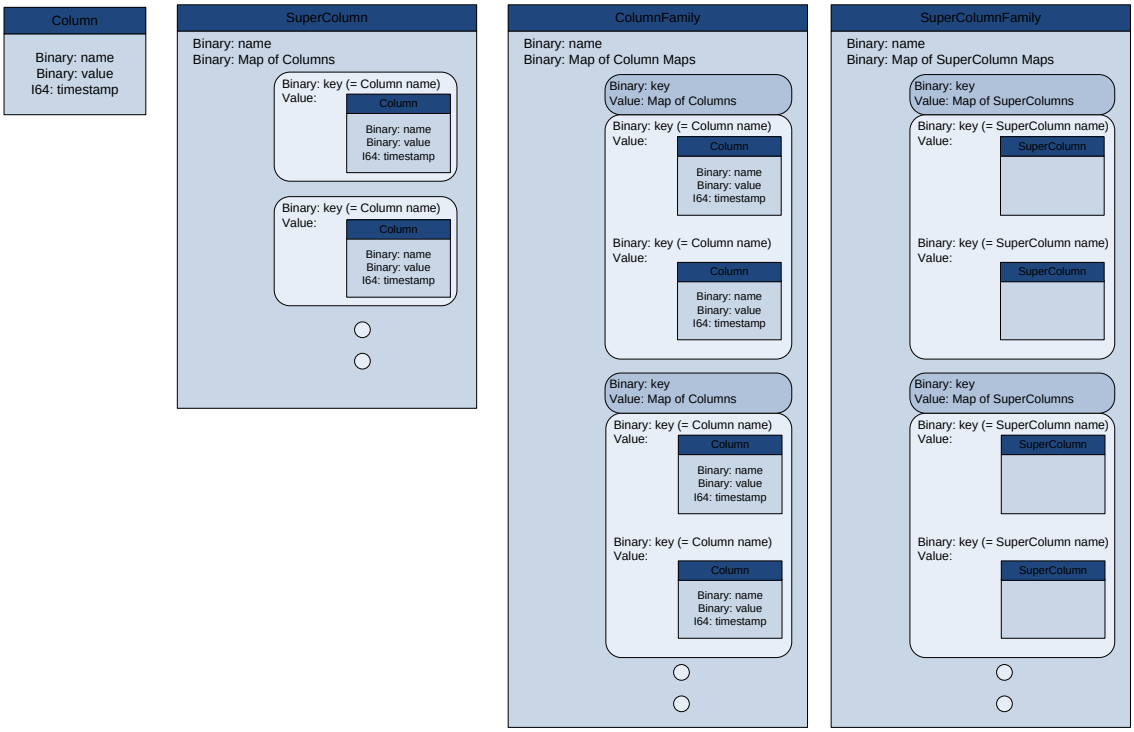

Fig. 61 Estructura de datos en Cassandra

Para realizar las configuraciones de almacenamiento de datos en Cassandra, es necesario editar el fichero storage-conf.xml. En este fichero se pueden modificar propiedades del clúster de nodos, como el nombre o el puerto de escucha; además de la definición estructura de la base de datos y como se ordenan estos datos.

Por defecto, Cassandra ordena los datos a medida que se van guardando en la base de datos. Esto proporciona un enorme rendimiento, pero sin embargo, hace que haya más trabajo antes de empezar a guardar datos. Existen 6 maneras diferentes de ordenar los datos dependiendo del tipo de variables:

- BytesType
- UTF8Type
- LexicalUUIDType
- TimeUUIDType
- AsciiUUIDType
- LongType

Mediante el tag CompareWith en la definición de una ColumnFamily o SuperColumnFamily se puede definir un orden de los datos dependiendo del tipo.

C.2. Arquitectura del clúster en Cassandra

La arquitectura de un sistema de almacenamiento que necesita trabajar en un entorno de producción es compleja. Además de la persistencia de datos, el sistema necesita asegurar una serie de características:

- Soluciones escalables y robustas ante el balanceo de carga.
- Adhesión de miembros y detección de fallos.
- Recuperación ante caídas.
- Sincronización de réplicas.
- Sobrecarga de manejo.
- Estado de transferencia.
- Concurrencia y planificación de tareas.
- Clasificación de solicitudes.
- Enrutamiento de solicitudes.
- Sistema de monitorización y alarma.
- Configuración de gestión.

En este caso, se explicarán los módulos que hacen de Cassandra un gran sistema de almacenamiento distribuido: particionado, replicación, adhesión de miembros, manejo de caídas y escalado del sistema. Todos ellos trabajan en sincronía para manejar correctamente las peticiones de lectura y escritura.

C.2.1. Particionado

Una de las claves de diseño de Cassandra es la capacidad de escalado incremental. Para ello es necesario, la partición de datos de manera dinámica en el conjunto de nodos del clúster.

A cada nodo del sistema, se le asigna un valor aleatorio que representa su posición dentro del anillo que representa el clúster. Cada elemento de datos identificado por una clave es asignado a un nodo haciendo un hash de la clave del elemento de datos cediendo su posición en el anillo, y moviéndose hacia la

derecha para encontrar el primer nodo con una posición mayor que la suya. Este nodo se considera el responsable de esa clave.

La aplicación especifica esta clave y Cassandra lo utiliza para enrutar las peticiones. Así, cada nodo se convierte en responsable de la región en el anillo entre ella y su nodo predecesor. La principal ventaja de este "hashing consistente", es que la llegada o la salida de un nodo sólo afectan a sus vecinos inmediatos y no a los otros nodos.

C.2.2. Replicación

Cassandra utiliza la replicación de datos para conseguir alta disponibilidad y durabilidad. Cada elemento de datos se replica en N nodos, donde N es el factor de replicación configurado. El coordinador se encarga de la replicación de los elementos de datos que caen dentro de su rango. Además, para almacenar localmente cada clave dentro de su rango, el coordinador replica estas claves en los N-1 nodos del anillo.

Cassandra define tres tipos diferentes de políticas de réplicas desde la aplicación:

- Rack Unaware: Se escogen N-1 sucesores del nodo coordinador de esos datos, para almacenar las réplicas de los datos.
- Rack Aware y Datacenter Aware: Son dos estrategias mucho más complejas. Cassandra elige un líder entre sus nodos utilizando un sistema llamado Zookeeper. Todos los nodos que se unen al clúster contactan con el líder que les asigna de qué rangos son responsables. Además el líder procura que ningún nodo sea responsable de más de N-1 rangos del anillo.

C.2.3. Adhesión de miembros y detección de fallos

En cuanto a la adhesión de miembros, Cassandra se basa en el protocolo Gossip. Este protocolo está inspirado en la propagación de chismes vistos en las redes sociales.

La detección de fallos es un mecanismo mediante el cual un nodo localmente puede determinar si otro nodo del clúster ha caído o continúa activo. En Cassandra la detección de fallos se utiliza para evitar intentos de comunicación con nodos que son inalcanzables durante varias operaciones. Para ello, Cassandra utiliza una versión modificada del ϕ Detector de Fallos Acumulativo. La idea principal de este método es que el detector de fallos no devuelve un valor booleano de si el nodo está activo o no; sino que devuelve un valor que representa el nivel de sospecha de cada nodo monitorizado.

Este valor es definido como ϕ y se expresa en una escala que se ajusta dinámicamente para reflejar la red y las condiciones de carga de los nodos monitorizados.

C.2.4. Arranque de nodos

Cuando un nodo arranca por primera vez, escoge un token que es equivalente a su posición en el anillo. La información del token es transmitida por todo el clúster, de esta manera, todos los nodos saben de su posición y la del resto de nodos en el anillo. Esto permite que cualquier nodo pueda enrutar una petición sobre una clave al nodo correcto.

En el momento del arranque, el nodo lee su fichero de configuración que contiene una lista con una serie de puntos de acceso para unirse al clúster, que se denominan semillas del clúster.

Los fallos en los nodos pueden ser causados de diferentes maneras como fallos de discos, mal comportamiento de la CPU, fallo en el suministro eléctrico, fallo en la red, etc. Raramente, una caída de un nodo representa una salida definitiva del clúster y, por tanto, no resulta un rebalanceo en la asignación de la partición o la reparación de replicas inalcanzables. De la misma manera, un error manual podría ocasionar la puesta en marcha involuntaria de nuevos nodos de Cassandra. A tal efecto, cada mensaje contiene el nombre del clúster de cada instancia de Cassandra.
Si un error manual en la configuración provoca que un nodo intente unirse a una instancia de Cassandra incorrecta, puede ser frustrado basado en el nombre del clúster.

Por estas razones, se consideró apropiado utilizar un mecanismo explícito para iniciar la adición y eliminación de nodos de una instancia de Cassandra. Un administrador utiliza una herramienta de línea de comandos o un navegador para conectarse a un nodo de Cassandra y emitir un cambio de adhesión a unirse o abandonar el clúster.

C.2.5. Escalado del clúster

Cuando un nuevo nodo se añade al sistema, se le asigna un token que le permite aliviar a un nodo muy cargado. Esto se traduce en una nueva división de rangos de los que algunos nodos eran responsables anteriormente.

El algoritmo de arranque de Cassandra se inicia desde cualquier otro nodo en el sistema por un operador usando la herramienta de línea de comandos. La experiencia práctica muestra que los datos pueden ser transferidos a una tasa de 40 MB/s. Actualmente, se está trabajando para mejorar este aspecto, tratando de tener varias réplicas participando en la transferencia de arranque consiguiendo paralelizar el esfuerzo de manera similar al caso de Bittorrent.

D. OpenID

El proyecto OpenID aparece desde que las URIs son el núcleo fundamental de la arquitectura Web, ya que proporcionan una base muy solida para la identificación de usuarios. De manera que cualquiera se puede identificar en Internet a través de una URL/URI o XRI y ser verificado por cualquier servidor que soporte este protocolo.

El sistema OpenID fue desarrollado por Brad Fitzpatrick, David Recordon, Josh Hoyt y Dick Hardt. Desde 2007, existe una Fundación OpenID en los Estados Unidos y una filial en Europa llamada OpenID Europe, es una organización sin ánimo de lucro que se dedica a organizar y desarrollar el framework de OpenID y a administrar su propiedad intelectual.

D.1. Terminología

- **Usuario final**: la persona que quiere acceder con su identidad a un sitio web.

- **Idendificador**: la URL o XRI elegida por el usuario final como identificador OpenID.

- **Proveedor de identidad**: un proveedor de servicios que ofrece registro de URL o XRI OpenID y provee autenticación OpenID.

- **Parte confidente**: el sitio web que quiere verificar la identidad del usuario final.

D.2. Tipos de identificadores

- **URI/URL**. Existen dos formas de obtener una URL OpenID válida para registrarse. La primera opción es utilizar una URL existente que el usuario controle, como un blog propio, y editar el HTML para insertar los tags OpenID en el código siguiendo las especificaciones. La segunda manera es obtener un identificador OpenID de un proveedor de identidades, estos proporcionan URLs configuradas automáticamente para ser utilizadas con servicios de autenticación OpenID.
  Ejemplo: usuario.proveedor-openid.org

- **XRI**. Los XRIs son un nuevo sistema de identificación en Internet diseñados específicamente para identidades digitales. Existen dos tipos: los i-nombres y los i-números. Se registran como equivalentes, pero la diferencia es que los i-nombres son reasignables mientras que los i-números nunca pueden serlo. De esta forma, cuando se utiliza un i-nombre XRI como identificador OpenID, se resuelve inmediatamente por el i-número equivalente, que es finalmente el identificador OpenID almacenado en la web solicitante.
  Ejemplo i-nombre: ejemplo.usuario

D.3. Como funciona OpenID

La autenticación OpenID es un método que permite utilizar un único nombre de usuario en un proveedor de identidad de confianza para permitir automáticamente el acceso a otros sitios web, sin la necesidad de registrarse en cada uno de ellos. El nombre de usuario es una URI o un XRI, y su contraseña (u otros credenciales posibles) permanece almacenada de forma segura en un servidor OpenID, este proveedor puede crearlo el usuario o utilizar el de un tercero.

En los sitios web que soportan OpenID, no es necesario crear nuevas cuentas o complementos de registro para obtener acceso. Únicamente es necesario disponer de un identificador OpenID. Para conseguirlo, estas páginas colocan un formulario de registro que sólo dispone del campo para el identificador, al contrario de los típicos formularios de identificación, que solicitan un nombre de usuario y una contraseña.

Para completar una autenticación mediante OpenID, hay que seguir los siguientes pasos:

1. **Envío del identificador:** cuando un usuario quiere acceder a una página web (ejemplo.com), debe utilizar el identificador OpenID (usuario.proveedor-openid.org) que previamente ha registrado en el proveedor de identidad (proveedor-openid.org). En la autenticación OpenID se puede utilizar como identificador una URL (usuario.proveedor-openid.org) o un i-nombre de una XRI (ejemplo.usuario o ejemplo.comunidad*usuario).

   Si el identificador es una URL, la parte confidente ha de transformarla a una forma canónica (http://usuario.proveedor-openid.org). En la versión OpenID 1.0, la parte confidente descubre cual es la URL del proveedor donde tiene que continuar con el proceso de autenticación. (http://proveedor-openid.org/openid-auth.html). A partir de la versión 1.1, el cliente hace el descubrimiento de los datos solicitando el documento XRDS con el tipo de contenido application/xrds+xml, que puede estar disponible en la URL y siempre está a través de una XRI.

2. **Intercambio de un secreto compartido**: la parte confidente y el proveedor, opcionalmente, establecen un secreto compartido, que almacena la parte confidente. Una vez completado este paso, hay dos maneras de comunicarse con la parte confidente:

   - *checkid_immediate*: está orientado a la identificación automática y en la que toda comunicación entre los dos servidores se realiza como una tarea transparente, sin que el usuario sea consciente de ello. En este caso, se pasaría directamente al paso 6, saltándose la autenticación de usuario en el proveedor.

- *chekid_setup*: en este caso, el usuario se comunica directamente con el servidor proveedor utilizando el mismo navegador web que emplea para acceder a la página de la parte confidente.

La segunda opción, checkid_setup, es la más popular en Internet ya que el usuario se puede registrar y confirmar el acceso a sus datos de manera consciente. Aunque la opción checkid_immediate se puede acabar convirtiéndose en un checkid_setup si la operación no puede automatizarse.

3. **Redirección del usuario final hacia el proveedor:** si se emplea la opción checkid_setup, la parte confidente redirige el navegador web del usuario final hacia el proveedor para que este pueda autenticarse (ejemplo.com redirigiría al usuario a proveedor-openid.org).

4. **Autenticación del usuario en el proveedor**: el método de autenticación en el proveedor puede variar, pero lo más habitual es pedir un nombre de usuario y contraseña, y después almacenar la sesión utilizando cookies.

5. **Pregunta sobre la confianza del usuario final en la parte confidente:** una vez que el usuario tiene la sesión abierta en el proveedor (proveedor-openid.org), este le pregunta acerca de la confianza en la página designada por la parte confidente para recibir los datos del usuario (http://ejemplo.com/openid-retorno.html).

   Si el usuario decide no fiarse, el navegador se redirige a la página indicando el rechazo de manera que no se podrá autenticar al usuario en la parte confidente (ejemplo.com).

6. **Confirmación de la autenticidad del proveedor:** si se responde positivamente a la confianza, la autenticación con el proveedor OpenID se acaba. En este momento, la parte confidente (ejemplo.com) no puede decidir si las credenciales recibidas realmente fueron recibidas desde el proveedor autentico (proveedor-openid.org). Por esta razón, el consumidor valida el secreto recibido del proveedor con las credenciales que había guardado previamente.

7. **Envío de los datos de usuario y registro en la parte confidente:** la parte confidente ya recibe los datos de usuario desde el proveedor de identidad y se considera registrado en la página confidente (ejemplo.com). El sitio puede entonces guardar la sesión o, si este es su primer registro, pedir al usuario que introduzca información específica para terminar el registro.

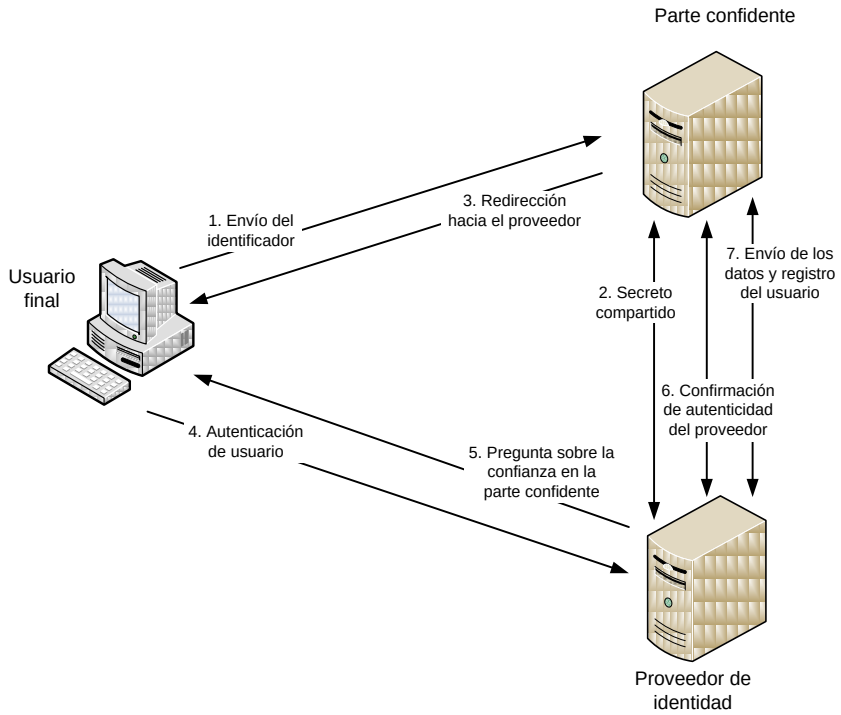

Fig. 62 Autenticación mediante OpenID

D.4. Seguridad

OpenID proporciona una arquitectura Single Sign-On, sin embargo no especifica el mecanismo de autenticación de usuario en el servidor OpenID, por lo que la seguridad de la conexión depende de la confianza que tenga el usuario final en el proveedor de identidad.

Algunos observadores sugieren que OpenID tiene debilidades de seguridad y puede resultar vulnerable a ataques de phishing. Por ejemplo, un usuario malintencionado podría dirigir al usuario final a una página de autenticación de identidad falsa indicando que introduzca sus credenciales. Una vez completado el proceso, la parte malintencionada podría tener acceso a la cuenta del usuario final con su proveedor de identidad y utilizar su cuenta OpenID para acceder a otros servicios.

Por esta razón, en un intento de combatir los posibles ataques de phishing, algunos proveedores de identidad obligan al usuario final a autenticarse primero con ellas antes de hacer la autenticación con la página web a la que se quiere acceder.

En Diciembre de 2008, la Fundación OpenID aprobó la versión 1.0 del Provider Authenticacion Policy Extension (PAPE), que permite a las partes confidentes pedir a los proveedores OpenID que empleen políticas de autenticación especificas al autenticar usuarios y que los proveedores informen a las partes confidentes sobre cuáles son las políticas que se están utilizando.

E. XMPP

Extensible Messaging and Presence Protocol, más conocido como XMPP (*Protocolo extensible de mensajería y comunicación de presencia*) (anteriormente llamado Jabber), es un protocolo abierto y extensible basado en XML, originalmente ideado para mensajería instantánea.

Con el protocolo XMPP queda establecida una plataforma para el intercambio de datos XML que puede ser usada en aplicaciones de mensajería instantánea. Las características en cuanto a adaptabilidad y sencillez del XML son heredadas de este modo por el protocolo XMPP.

A diferencia de los protocolos propietarios de intercambio de mensajes, como por ejemplo Windows Live Messenger, XMPP se encuentra documentado y se insta a utilizarlo en cualquier proyecto. Existen servidores y clientes libres que pueden ser usados sin coste alguno.

Este es el protocolo que seleccionó Google para su servicio de mensajería Google Talk.

E.1. Historia

Jeremie Miller comenzó el proyecto Jabber en 1998. Su primera liberación de software importante sucedió en Mayo de 2000. El principal producto del proyecto era jabberd, un servidor XMPP.

Este protocolo XMPP inicial creó las bases para el XMPP, publicado como RFC 3920. A menudo, ha sido considerado competidor de SIMPLE, basado en el protocolo SIP, como protocolo estándar de mensajería instantánea y notificación de presencia. Jabber Software Foundation fue renombrado como XMPP Standards Foundation el 15 de Enero de 2007.

A partir de 2005, existían cerca de una media docena de implementaciones de servidores XMPP, escritos en diferentes lenguajes de programación.

En Agosto de 2005, Google introdujo Google Talk, una combinación de VoIP y sistemas de gestión de identidades que usa XMPP para sus funciones de mensajería instantánea y como base para su protocolo de señalización de transferencias y su sistema de voz. El lanzamiento inicial no incluía comunicaciones de servidor a servidor, pero el 17 de Enero de 2006, fueron añadidas.

E.2. Ventajas

- *Descentralización*

  La arquitectura de las redes XMPP es similar a la del correo electrónico; cualquiera puede poner en marcha su propio servidor XMPP, sin que haya ningún servidor central.

- *Estándares abiertos*

La Internet Engineering Task Force ha formalizado el protocolo XMPP como una tecnología de mensajería instantánea estándar, y sus especificaciones han sido publicadas como los RFC 3920 y RFC 3921. El desarrollo de esta tecnología no está ligado a ninguna empresa en concreto y no requiere el pago de royalties.

- *Seguridad*

Los servidores XMPP pueden estar aislados de la red pública XMPP, y poseen robustos sistemas de seguridad. Para apoyar la utilización de los sistemas de cifrado, la XMPP Standards Foundation pone a disposición de los administradores de servidores XMPP Autoridad de certificación en xmpp.net ofreciendo certificados digitales gratis.

- *Flexibilidad*

Se pueden hacer funcionalidades a medida sobre XMPP; para mantener la interoperabilidad, las extensiones más comunes son gestionadas por la XMPP Software Foundation.

E.3. Desventajas

- *Sobrecarga de datos de presencia*

Típicamente cerca de un 70% del tráfico entre servidores son datos de presencia, y cerca de un 60% de estos son transmisiones redundantes. Actualmente se están estudiando nuevos protocolos para aliviar este problema.

- *Escalabilidad*

XMPP también sufre el mismo problema de redundancia en los servicios de chatroom y de suscripción. Actualmente se está trabajando en su solución.

- *Sin datos binarios*

XMPP es codificado como un único y largo documento XML, lo que hace imposible entregar datos binarios sin modificar. De todas formas, las transferencias de archivos se han solucionado usando otros protocolos como HTTP. Si es inevitable, XMPP también puede realizar transferencias codificando todos los datos mediante base64.

E.4. Descentralización y direccionamiento

La red XMPP está basada en servidores, pero descentralizada; por diseño, no hay ningún servidor central, como sucede con servicios como AOL Instant Messenger o MSN Messenger. Sobre este punto, surge cierta confusión, puesto que existe un servidor XMPP público en "Jabber.org", al que están suscritos un gran número de usuarios, pero no hay que olvidar que cualquiera puede poner en marcha su propio servidor. El puerto estandar utilizado para XMPP es el 5222.

Cada usuario en la red XMPP tiene un único identificador (*Jabber ID*, normalmente abreviado como *JID*). Para evitar la necesidad de un servidor central con una lista exhaustiva de identificadores, el *Jabber ID* está estructurado como una dirección de correo electrónico, con un nombre de usuario y una dirección DNS para el servidor en el que reside el usuario, separado por un signo @. Un identificador Jabber sería algo como *nombredeusuario@dominio.com*.

Como un usuario puede querer identificarse desde distintos lugares, el servidor permite al cliente especificar una cadena de referencia conocida como recurso, que identifica el cliente que está utilizando el usuario (por ejemplo: casa, trabajo, portátil, etc.). Esto será incluido en el JID añadiendo un carácter / seguido del nombre del recurso. Cada recurso debe tener especificada un valor numérico de prioridad. Por ejemplo el JID completo de la cuenta del trabajo del usuario sería: *nombredeusuario@dominio.com/trabajo*.

Los mensajes de la forma *nombredeusuario@dominio.com* serán dirigidos al cliente con mayor prioridad, y los de la forma *nombredeusuario@dominio.com/trabajo* serán dirigidos al cliente del trabajo.

Los JID sin la parte del nombre de usuario también son válidos y se utilizan para enviar mensajes de sistema y control.

F. SERVIDOR OPENFIRE

El servidor Openfire es un sistema de mensajería instantánea con licencia Open Source GPL. Esta hecho en Java y utiliza el protocolo XMPP.

Openfire implementa las siguientes características:

- SSL/TLS
- Interfaz amigable
- Adaptable según las necesidades
- Estadísticas del Servidor, mensajes, paquetes, etc.
- Clúster con múltiples servidores
- Transferencia de Archivos
- Compresión de datos
- Mensajes offline
- Favoritos
- Autenticación vía Certificados u otros métodos seguros como: Kerberos, LDAP, PAM y Radius
- Almacenamiento en diferentes tipos de sistemas de almacenamientos de datos como: Active Directory, LDAP, MS SQL, MySQL, Oracle y PostgreSQL
- Conferencias

La administración del servidor se hace a través de una interfaz web, que corre por defecto en el puerto 9090 (HTTP) y en el puerto 9091 (HTTPS). Gracias a este servicio, los administradores pueden conectarse a la web y editar la configuración del servidor de forma rápida y sencilla.

A través de la interfaz de administración se puede:

- Administrar usuarios
- Administrar grupos
- Crear salas de conferencias permanentes
- Añadir plugins
- Editar la configuración del almacenamiento de datos

G. Adobe FLEX

Adobe Flex, hasta 2005 llamado Macromedia Flex, es un término que agrupa una serie de tecnologías publicadas desde Marzo de 2004 por Macromedia para dar soporte al despliegue y desarrollo de Aplicaciones Enriquecidas de Internet, basadas en su plataforma propietaria Flash.

Los programadores tradicionales de aplicaciones ven como un desafío adaptar la metáfora de la animación sobre la plataforma con la cual fue originalmente construido Flash. Flex minimiza elegantemente este problema proveyendo un flujo de trabajo y un modelo de programación que es familiar a los desarrolladores de aplicaciones.

Flex fue inicialmente liberado como una aplicación de la J2EE o biblioteca de etiquetas JSP que compilara el lenguaje de marcas Flex (MXML) y ejecutara mediante ActionScript aplicaciones Flash (archivos SWF binarios). Versiones posteriores de Flex soportan la creación de archivos estáticos que son compilados, y que pueden ser distribuidos en línea sin la necesidad de tener una licencia de servidor.

Flex pone en relieve el desarrollo de Interfaces gráficas de usuario usando un lenguaje XML llamado MXML. Flex tiene varios componentes y características que aportan funcionalidades tales como Servicios Web, objetos remotos, arrastrar y soltar, columnas ordenables, gráficas, efectos de animación y otras interacciones simples. El cliente solo carga la aplicación una vez, mejorando así el flujo de datos frente a aplicaciones basadas en HTML (como por ejemplo PHP, ASP, JSP), las cuales requieren de ejecutar plantillas en el servidor para cada acción. El lenguaje y la estructura de archivos de Flex buscan el desacoplamiento de la lógica y el diseño.

El servidor Flex también actúa como un gateway permitiendo al cliente comunicarse con servicios web XML y objetos remotos (tales como Coldfusion CFCs, clases Java, y cualquiera que soporte el formato de mensajes de acciones).

H. Algoritmo de búsqueda A*

Este algoritmo es un algoritmo de optimización de cambios de estado. En él se examina un sistema en su estado actual y se determina la serie de cambios de estado de menor coste necesarios para llegar a un estado final. Se suele utilizar para encontrar un camino entre nodos, pero también puede ser usado para resolver el Cubo de Rubik o puzles deslizantes.

El algoritmo A* utiliza una función de evaluación $f(n) = g(n) + h'(n)$:

- $h'(n)$ representa el valor heurístico del nodo a evaluar desde el actual, n, hasta el final.
- $g(n)$ representa el coste real del camino recorrido para llegar a dicho nodo, n.

A* mantiene dos estructuras de datos auxiliares, que podemos denominar *abiertos*, implementado como una cola de prioridad (ordenada por el valor $f(n)$ de cada nodo), y *cerrados*, donde se guarda la información de los nodos que ya han sido visitados. En cada paso del algoritmo, se expande el nodo que esté primero en abiertos, y en caso de que no sea un nodo objetivo, calcula la $f(n)$ de todos sus hijos, los inserta en abiertos, y pasa el nodo evaluado a cerrados.

El algoritmo es una combinación entre búsquedas del tipo primero en anchura con primero en profundidad: mientras que $h'(n)$ tiende a primero en profundidad, $g(n)$ tiende a primero en anchura. De este modo, se cambia de camino de búsqueda cada vez que existen nodos más prometedores.

A* cumple una serie de propiedades que lo hacen un algoritmo de búsqueda potente:

- Como todo algoritmo de búsqueda en anchura, A* es un algoritmo completo: en caso de existir una solución, siempre dará con ella.
- Si para todo nodo n del grafo se cumple $g(n) = 0$, nos encontramos ante una búsqueda voraz. Si para todo nodo n del grafo se cumple $h(n) = 0$, A* pasa a ser una búsqueda de coste uniforme no informada.
- Para garantizar la optimalidad del algoritmo, la función $h(n)$ debe ser admisible, es decir, que no sobrestime el coste real de alcanzar el nodo objetivo.
- De no cumplirse dicha condición, el algoritmo pasa a denominarse simplemente A, y a pesar de seguir siendo completo, no se asegura que el resultado obtenido sea el camino de coste mínimo. Asimismo, si garantizamos que $h(n)$ es consistente, es decir, que para cualquier nodo n y cualquiera de sus sucesores, el coste estimado de alcanzar el objetivo desde n no es mayor que el de alcanzar el sucesor más el coste de alcanzar el objetivo desde el sucesor.